UNIVERSITÄT BIELEFELD

Fakultät für Linguistik und Literaturwissenschaft

Dissertation
zur Erlangung des akademischen Grades
Doctor philosophiae (Dr. phil.)

Auszeichnungssprachen für linguistische Korpora

Theoretische Grundlagen, De-facto-Standards, Normen

Maik Stührenberg

24. April 2012

UNIVERSITÄT BIELEFELD

Fakultät für Linguistik und Literaturwissenschaft

Dissertation
zur Erlangung des akademischen Grades
Doctor philosophiae (Dr. phil.)

Auszeichnungssprachen für linguistische Korpora

Theoretische Grundlagen, De-facto-Standards, Normen

Maik Stührenberg

24. April 2012

Betreuer:
Dr. Andreas Witt
Prof. Dr. Dieter Metzing

Gutachter:
Prof. Dr. David Schlangen
Prof. Dr. Dieter Metzing

Gesetzt mit LaTeX (KOMA-Script und BibLaTeX aus TeXLive 2011) unter MacOS X 10.7.3.
Gedruckt auf alterungsbeständigen Papier nach ISO 9706.

Für Katrin, Kieran und Luca.

Erklärung

Hiermit erkläre ich die vorliegende Dissertation selbständig verfasst zu haben, keine anderen als die angegebenen Hilfsmittel verwendet und wörtlich und inhaltlich übernommene Quellen als solche gekennzeichnet zu haben.

Bielefeld, den 24. April 2012

Danksagung

Diese Arbeit hat profitiert von einer ganzen Reihe Personen, die an dieser Stelle nicht alle aufgeführt werden können. Namentlich genannt werden sollen hier allerdings die (ehemaligen) Kollegen, denen ich einige sehr hilfreiche Kommentare zu verdanken habe – auch wenn sie streckenweise aufwändige Überarbeitungen zur Folge hatten. Zu diesen Personen gehören (in alphabetischer Reihenfolge) Nils Diewald, Oliver Schonefeld und Christian Wurm. Darüber hinaus hat mir Andy Lücking immer wieder zahlreiche Hinweise zur Optimierung von LaTeX geben können, was einige Zeit bei Fehlersuche und Formatierung erspart hat.

Zu guter Letzt danke ich meiner Familie für die fortwährende Unterstützung und meinen Betreuern und Gutachtern für die wertvollen Anmerkungen sowie die Geduld, mit der sie auf die Fertigstellung dieser Arbeit gewartet haben.

Inhaltsverzeichnis

Das Zauberwort für die nächsten Jahre lautet Standardisierung.
Carstensen (2003, S. 160)

1 Einleitung

Das Zitat am Kopf dieser Seite stammt aus einer von Kai-Uwe Carstensen im Jahr 2003 in der Zeitschrift *Linguistik Online* verfassten Rezension. Der Autor beschreibt auch in den weiteren Worten treffend die Gefahren einer nicht-standardisierten Arbeitsweise in der Linguistik (und speziell in der Computerlinguistik):

> Ohne gemeinsame Bemühungen, Dinge auf einen gemeinsamen Nenner zu bringen und Gleiches beim selben Namen zu nennen, wird die Zukunft der CL von Verzettelung und Redundanz geprägt sein, und die Lösung zentraler Probleme wird weiter auf sich warten lassen. Dies gilt für alle relevanten Ebenen (Theorie, Methode, Implementation). (Carstensen 2003, S. 160)

Auf den ersten Blick erscheinen wissenschaftliches Arbeiten und Standards nur schwer miteinander vereinbar – auf der einen Seite das Streben nach neuen Erkenntnissen und Methoden, auf der anderen Seite das Zementieren eines nach oft mühsamen Verhandlungen ausgearbeiteten Kompromisses, der kreative Abweichungen davon unmöglich macht. Doch diese Sichtweise täuscht. Die Verwendung von standardisierten Verfahren und Formaten wird nicht nur durch wissenschaftliche Disziplinen wie die Texttechnologien in hohem Maße voran getrieben, ja oft erst ermöglicht, auch profitieren linguistische Anwendungen im hohen Maße von Standards. Darüber hinaus lässt sich ein Trend hin zu generischen Formaten verzeichnen, der Wissenschaftlern zumindest gewisse Freiheiten zubilligt (zur Diskussion der Vor- und Nachteile sei auf Kapitel 16 verwiesen).

Interessanterweise ist die Reichweite von standardisierten Formaten in den betreffenden Zielgruppen unterschiedlich groß: Während Akteure in der Industrie die Nutzung von Standards als Schlüssel zur effizienten und nachhaltigen, da interoperablen Nutzung von Ressourcen und Produkten begreifen (man denke nur an das Format DIN A4), sind potentielle Nutzer in der Wissenschaft oft noch nicht einmal darüber informiert, dass standardisierte Spezifikationen zu vielen Themen existieren, die sie direkt oder

indirekt betreffen. Dieser Umstand ist nicht zuletzt auch den an der Erarbeitung solcher Standards beteiligten Akteuren bekannt. So bedauern zwei in vielerlei Hinsicht in diesem Feld äußerst aktive Beteiligte:

> At present, language professionals and standardization experts are not sufficiently aware of the standardization efforts being undertaken by ISO/TC 37/SC 4. Promoting awareness of future activities and rising problems, therefore, is crucial for the success of the committee, and will be required to ensure widespread adoption of the standards it develops. (Ide und Romary 2004b, S. 211)

Gerade zum aktuellen Zeitpunkt erscheint eine solche Untersuchung relevant, da nach Jahren der Projektarbeit (eine Übersicht der 1990er Jahre findet sich beispielsweise in Cole *et al.* 1997) die Relevanz *nachhaltiger* Forschungsarbeit immer mehr an Bedeutung gewinnt. Die Linguistik als Disziplin beginnt gerade damit, sich mit Standards zu befassen und darauf einzulassen. So führt Simons (2007) Beispiele für Standards an, die im linguistischen Umfeld bereits entstanden sind. Neben ISO 639-3:2007 (vgl. auch die Ausführungen in Dalby *et al.* 2004) sind hier vorrangig UNICODE (Unicode 1.0; Unicode 6.0.0; ISO/IEC 10646:2011) und die GENERAL ONTOLOGY FOR LINGUISTIC DESCRIPTION (GOLD, vgl. Farrar und Langendoen 2003) zu nennen. Gerade die Einigung auf UNICODE als universellen Zeichensatz und die Festlegung auf UTF-8 als Vorgabe-Zeichenkodierung in XML machen mehrsprachige (und dennoch austauschbare) linguistisch annotierte Korpora möglich.[1] Aktuelle Entwicklungen betreffen vorrangig die Arbeiten in ISO/IEC TC 37/SC 4, die in Projekten wie „Common Language Resources and Technology Infrastructure“ (CLARIN) evaluiert werden (CLARIN 2009b).

1.1 Linguistische Korpora

Interessanterweise ist gerade bei einem wesentlichen Werkzeug der Sprachforschung, den linguistisch annotierten Korpora, erst relativ spät eine Tendenz hin zu einer Vereinheitlichung von Formaten zu beobachten – und dass, obwohl gerade hier die Speicherung, Annotation und Verarbeitung linguistischer Daten durch den Einsatz von standardisierten Formaten und Werkzeugen deutlich leistungsfähiger erfolgen kann, und die Bedeutung von Korpora für die Linguistik wieder stark zunimmt: „Linguistically-annotated corpora are an increasingly critical resource for research in linguistics and computational linguistics.“ (Ide 2007, S. 3)

Standards können dabei ein probates Mittel sein, um die Kosten der Korpuserstellung zu begrenzen und nachhaltige Korpusnutzung zu ermöglichen (Maxwell 2010). Waren die Anfänge der Erstellung linguistischer Korpora noch geprägt von einer gewissen Gedankenlosigkeit – sowohl in Bezug auf das physische Datenformat, in dem die Sprachdaten digital gespeichert wurden, als auch in Bezug auf die Art und Weise der strukturierten Auszeichnung, der Annotation (vgl. Ide 2007, S. 4), entsteht

[1] Gippert (2006) veranschaulicht ausführlich den Weg von 7- und 8-Bit-Kodierungen über Unicode bis hin zu XML-Annotationen.

jetzt, nach einer ganzen Reihe von Projekten, eine neue Wissenschaft, die sich mit der Entwicklung präziser Methoden und Werkzeuge zur Korpuserstellung befasst (vgl. Francopoulo, Declerck, Monachini *et al.* 2006). Angetrieben wird diese Entwicklung durch mehrere Faktoren: Zum einen stehen seit der Verwendung maschineller Annotation und Datenspeicherung in den 1980er Jahren immer mächtigere Hard- und Software zur Verarbeitung auch großer Korpora zur Verfügung. Zum anderen wurde mit der Einführung der Metasprachen STANDARD GENERALIZED MARKUP LANGUAGE (SGML, ISO 8879:1986) und der EXTENSIBLE MARKUP LANGUAGE (XML, Bray, Paoli und Sperberg-McQueen 1998; Bray, Paoli, Sperberg-McQueen, Maler und Yergeau 2004) der Grundstein für eine vereinfachte Verbreitung annotierter Daten gelegt (vgl. auch die Ausführungen in Abschnitt 3.2 dieser Arbeit). Der rasante Durchbruch von XML und dessen begleitenden Spezifikationen, darunter die vereinfachte Adressierung von annotierten und nicht-annotierten Daten mittels der XML PATH LANGUAGE (XPATH, Clark und DeRose 1999; Berglund, Boag *et al.* 2007), XML LINKING LANGUAGE (XLINK, DeRose, Maler und Orchard 2001; DeRose, Maler, Orchard und Walsh 2010) und XPOINTER (DeRose, Maler und Daniel 2002; Grosso *et al.* 2003b; Grosso *et al.* 2003a; DeRose, Daniel, Maler *et al.* 2003), die Transformation mittels XSLT (Clark 1999; Kay 2007), das Retrieval mit Hilfe von XQuery (Boag *et al.* 2007; Boag *et al.* 2010), und die Entwicklung neuer *Constraint Languages* (CL) wie XML SCHEMA-BESCHREIBUNGEN (XML SCHEMA DESCRIPTIONS, XSD, oder auch kurz: XML SCHEMA, Walmsley 2002; Fallside 2001; Fallside und Walmsley 2004; Thompson *et al.* 2001; Thompson *et al.* 2004; Biron und Malhotra 2001; Biron und Malhotra 2004) oder RELAX NG (Regular Language Description for XML New Generation, vgl. ISO/IEC 19757-2:2003; van der Vlist 2003b; ISO/IEC 19757-2:2008), hat weitreichende Implikationen auf die Möglichkeit der Speicherung linguistischer Korpora mit sich gebracht.[2] Mit zunehmend wachsender Akzeptanz standardisierter Metasprachen steigt auch der Bedarf nach Annotationsformaten, die einen Austausch der so ausgezeichneten Korpora ermöglichen, weshalb mehr und mehr Wissenschaftler weltweit an eben solchen standardisierten Austauschformaten arbeiten. Wie Ide (2007) ausführt, ist das endgültige Ziel dieser Bemühungen ein universelles Datenformat, das sowohl verschiedene linguistische Phänomene strukturiert speichern kann als auch Korpusdaten unterschiedlichster Sprachen verarbeiten kann.

Auch von anderer Seite kann eine verstärkte Nachfrage von standardisiert annotierten Korpora beobachtet werden: Wie Ide (2006) argumentiert, verlangt nicht nur die explosionsartige Verbreitung von Webtechnologien, sondern auch die stetig wachsende Masse an elektronischen Dokumenten, die in der Industrie verwaltet und verarbeitet werden müssen, nach linguistischen – oder explizit texttechnologischen – Methoden und Anwendungen. Insofern fällt die Verfügbarkeit von Ressourcen und texttechnologischen Methoden auf der einen Seite zusammen mit einer steigenden Nachfrage nach nachhaltig nutzbaren linguistischen Daten auf der anderen Seite.

Untersuchungsgegenstand dieser Arbeit sind standardisierte Annotationsverfahren zur Speicherung und Anreicherung von Korpusdaten. Damit rückt auch der Begriff des Korpus in den Mittelpunkt dieser Arbeit, der von Sinclair (2005, S. 16) wie folgt definiert wird: „A corpus is a collection of pieces of language text in electronic form, selected

[2] In Anfängen wird diese Entwicklung bereits in Ide (2000) dargelegt.

according to external criteria to represent, as far as possible, a language or language variety of data for linguistic research.“ Zu dieser Definition kommt der Autor als Quintessenz aus zehn Leitprinzipien zur Erstellung von Korpora. Himmelmann (2006) fügt dieser Definition noch die Punkte Dauerhaftigkeit (*lasting*) und Vielseitigkeit bzgl. der Verwendung (*multipurpose*) hinzu, ähnliche Definitionen finden sich in McEnery *et al.* (2006a). Die grundlegenden Anforderungen an einen Korpus können daher folgt formuliert werden:

1. Ein Korpus (von Witt, Heid *et al.* 2009, S. 2, auch als *Static Text-based Language Ressources* bezeichnet) ist eine elektronische (oder auch: digitale) Sammlung linguistischer Daten verschiedenen Ursprungs (Schriftsprache, gesprochene Sprache, digitale Kommunikation, um nur einige Beispiele zu nennen). Im Allgemeinen bestehen Textkorpora aus geschriebenen oder transkribierten gesprochenen Texten. Im Gegensatz zu Sinclair (2005) wird für diese Arbeit der Korpusbegriff erweitert um weitere digitale Medien wie Audio- und Videodaten, die neben dem Text Untersuchungsgegenstand linguistischer Betrachtungen sein können (vgl. auch Himmelmann 2006).

2. Die Auswahl an linguistischen Daten erfolgt nach (text-)externen Kriterien (d. h., anhand der kommunikativen Funktion).

3. Die Korpusdaten sollten nach – im Rahmen der Möglichkeiten – eine Sprache oder Sprachvarietät repräsentieren.

4. Die Speicherung der Korpusdaten sollte in einem Format erfolgen, das eine nachhaltige und universelle Nutzung ermöglicht.

Als Teildisziplin der Sprachwissenschaft befasst sich die Korpuslinguistik mit Fragen der Erstellung, Speicherung und Nutzung von Korpora.[3] Wie Meyer (2008) ausführt, lassen sich die Anfänge der Korpuslinguistik in die prä-digitale Zeit zurück verfolgen, bis ins 18. Jahrhundert und noch weiter. Als besonders einflussreiches Beispiel eines nicht-digitalen Korpus aus dem 20. Jahrhundert führt Meyer den „Survey of English“ (SEU) Korpus (1956, vgl. Meyer 2008, S. 10f.). Relevanter für die vorliegende Arbeit sind die elektronisch erfassten Sprachkorpora, deren Ursprünge Johansson (2008) folgend in den 1960er Jahren begannen (der erste mit Computerhilfe erfasste Textkorpus, der „Brown University Standard Corpus of Present-Day American English“, oder kurz: Brown-Korpus, wurde 1964 an der Brown University, Providence, Rhode Island, erstellt). Insgesamt wurde eine Reihe von Formaten zur Speicherung von Korpora im Laufe der letzten fast 50 Jahre entwickelt. Im deutschsprachigen Raum ist hier sicherlich das NeGra-Format zu nennen (Brants 1997), das in der gleichnamigen Baumbank und als Exportformat auch für das TIGER-Korpus Verwendung findet, aber natürlich auch die vom *Institut für deutsche Sprache* (*IDS*) in Mannheim verwalteten Korpora, darunter das DEutsche REferenzKOrpus (DEREKO, für aktuelle Entwicklungen sei verwiesen auf Kupietz, Belica *et al.* 2010; Kupietz, Schonefeld *et al.* 2010).

[3] Zur Frage der Nutzung kann exemplarisch auf Garside *et al.* (1997) verwiesen werden, der exemplarisch folgende Annotationsebenen auf Basis der gesammelten linguistischen Daten anführt (ohne Anspruch auf Vollständigkeit): Phonetik, Prosodie, Syntax, Semantik, Pragmatik, Diskurs, Stil und Lexik.

Diese Arbeit konzentriert sich auf den ersten und letzten Punkt der vorherigen Aufzählung: Die elektronische Speicherung von linguistischen Daten, die aus digitalisierten Text-, Audio- oder Videoströmen bestehen können, und nach verschiedenen linguistischen Kriterien untersucht werden können und deren Speicherung in einem nachhaltigen und universell einsetzbaren Format. Gerade letzter Punkt ist eng verbunden mit der Frage der Annotation.

1.2 Annotation und Primärdaten

Als Annotation wird prinzipiell die Auszeichnung von Primärdaten mit Anmerkungen verstanden (vom lateinischen *annotatio*: schriftliche Bemerkung, Anmerkung, Aufzeichnung, vgl. Wahrig 1996), also die Anreicherung der zu annotierenden Daten mit Informationen. Als Primärdaten werden die zu annotierenden Daten (in textueller, digitaler Form) angesehen (Ide 1998, S. 465). Davon trennen lassen sich wiederum die Rohdaten, also die Informationsobjekte in ihrer ursprünglichen Form.[4] Im Falle von nicht-digital (oder nicht in Textform) vorliegenden linguistischen Rohdaten (z. B. bei der Erstellung diachroner oder multi-modaler Korpora) werden zunächst Transkriptionen angefertigt, die anschließend als Primärdatum dienen.

Während in Ide (1998, S. 465) davon die Rede ist, dass Primärdaten auch durchaus solche Dateien sein können, die bereits in einem digitalen Format aufbereitet und mit Formatierungsanweisungen angereichert sein können (vgl. Abschnitt 7.2), fassen manche in dieser Arbeit diskutierten Spezifikationen (beispielsweise das in Teil III diskutierte XStandoff) den Begriff enger und verlangen als Primärdatum eine vollständig unannotierte, d. h. in reiner Textform (TXT-Datei) vorliegende Repräsentation des Textes. Als Beispiel für eine Auszeichnungsebene findet sich in Leech (2005, S. 17) eine exemplarische Part-of-Speech-Annotation:

> *present*_NN1 (singular common noun)
> *present*_VVB (base form of a lexical verb)
> *present*_JJ (general adjective)

Es ist in der Linguistik nicht unumstritten, ob diese (oder eine andere) Form der Annotation direkt an die Primärdaten angereichert (und damit unter Inkaufnahme einer Veränderung der Originaldaten in der gleichen Datei) oder separat gespeichert werden soll (vgl. Sinclair 2005, vs. Leech 2005). Im ersten Fall spricht man von *Inline*-, im zweiten von *Standoff*-Annotation (vgl. Abschnitt 3.3.2.3). Beide Konzepte finden bei den in dieser Arbeit vorgestellten Spezifikationen Eingang und für beide Vorgehensweisen gibt es sowohl Argumente, die dafür als auch dagegen sprechen: Werden die Rohdaten nicht verändert, ist es problemlos möglich, sie als Ausgangspunkt für verschiedene Annotationsebenen (vgl. Kapitel 3) zu verwenden. Des Weiteren besteht die Gefahr, dass eine vorhandene Annotation die Sicht auf den zu untersuchenden Gegenstand

[4] Zur Unterscheidung sei beispielsweise auf Himmelmann (1998) oder auch Lemnitzer und Zinsmeister (2006, S. 46f.) verwiesen. Zu beachten ist, dass der Sprachgebrauch in dieser Hinsicht nicht einheitlich ist: So bezeichnet Himmelmann (2006) als Primärdaten die unannotierten Informationsobjekte, für die im weiteren Verlauf dieser Arbeit der Term „ Rohdaten“ verwendet wird.

verschleiert, da sie die Sichtweise des Annotators widerspiegelt. Dagegen (und für eine direkte Auszeichnung der Primärdaten) spricht, dass eine oder mehrere Annotationen zusätzliche Informationen über einen Text speichern, die im einfachsten Fall zwar für einen geeigneten Rezipienten explizit sichtbar sind, in schwereren Fällen aber nur implizit visualisiert werden können, und auf deren Basis erst weitere Untersuchungen möglich sind.

Zusammenfassend nennt Leech (1993, S. 275) folgende Maximen der Korpusannotation:

1. Annotationen sollten sich rückstandslos aus den Korpusdaten entfernen lassen, um jederzeit unannotierte Informationsobjekte wiederherstellen zu können.

2. Damit verbunden sollte es möglich sein, Annotation und Primärdatum getrennt zu speichern (entweder in separaten Dateien oder in zumindest getrennten Zeilen).

3. Das Annotationsschema sollte in Form von Richtlinien dokumentiert sein. Diese sollten neben einer Übersicht über das Annotationsinventar auch die Definition der einzelnen Einheiten und Hinweise zur Verwendung enthalten.

4. Es sollte dokumentiert sein, wie und von wem die Annotation durchgeführt wurde.

5. Das Annotationsschema sollte neutral gehalten sein (z. B. in Bezug auf linguistische Theorien), damit es auch in anderen Annotationsvorhaben eingesetzt werden kann („Caveat Emptor Principle“). Keinesfalls sollte es als final und unfehlbar bezeichnet sein.

6. Keinesfalls kann ein Annotationsschema a priori als Standard gelten, da Standards durch die entsprechende breite Nutzung entstehen.

Ein Beispiel für ein international sehr etabliertes Annotationsformat ist die Klammernotation der Penn-Treebank (Marcus *et al.* 1993). Allerdings ist unbestritten, dass seit der Entwicklung von standardisierten Metasprachen wie SGML und XML die Annotation mehrheitlich direkt auf den zu annotierenden Daten erfolgt (sofern diese in digitaler Textform vorliegen; eine Ausnahme bildet die bereits erwähnte *Standoff*-Annotation). Analog zu dem oben angegebenen Beispiel könnte daher eine Auszeichnung in XML wie folgt aussehen:

Listing 1.1: Mögliche XML-Annotation

```
1 <noun>present</noun>
2 <verb>present</verb>
3 <adjective>present</adjective>
```

Dabei ist die Festlegung auf eine gemeinsame Syntax, in Form einer standardisierten Metasprache, nur ein (allerdings wesentlicher) Teil zur Entwicklung austauschbarer Korpusdaten. Das Vokabular selbst, also die zur Verfügung stehenden Elemente und Attribute, bzw. die Verschachtelung der Elemente, ist Gegenstand langer Diskussionen und Standardisierungsbemühungen. Einige Ergebnisse dieser Diskussionen werden in den folgenden Kapiteln dieser Arbeit thematisiert. Ein besonderes Augenmerk liegt

dabei auf der Unterstützung multipler Annotationen (Mehrebenen-Annotation), da diese immer stärker in den Fokus wissenschaftlicher Forschung rückt.[5]

1.3 Gegenstand und Ziele der Arbeit

Ein Ziel der vorliegenden Arbeit ist es, die Spannweite von linguistisch motivierten Standards deutlich zu machen, diese vorzustellen und zu bewerten und damit dazu beizutragen, dass standardisierte Formate zukünftig auch in der Wissenschaft eine größere Verbreitung erreichen als es bisher der Fall ist. Zwar finden sich bereits Gegenüberstellungen verschiedener Annotationsschemata (beispielsweise in Leech 1993; Sasaki und Witt 2004; Ule und Hinrichs 2004; Leech 2005; McEnery *et al.* 2006b), diese thematisieren aber nicht die aktuellen Entwicklungen im Bereich der Normierung. Der Teil II dieser Arbeit soll daher maßgebliche Spezifikationen vorstellen und deren Einsatzmöglichkeiten diskutieren, wobei der Schwerpunkt auf textuellen Primärdaten liegt, d. h., Formaten zur Annotation von multimodalen Daten werden nicht behandelt. Zu beachten ist, dass nicht nur internationale Normen behandelt werden, sondern auch De-facto-Standards und *Best Practices* (zur Begriffsdefinition vgl. Kapitel 2). Daran schließt sich die Präsentation des eigenen Forschungsansatzes in Form der Metasprache XStandoff im Teil III an. Auch wenn diese nicht im Rahmen einer Normierungsarbeit entstanden ist, wurde sie doch zum gleichen Zweck entwickelt und kann als Best Practice-Implementierung angesehen werden. XStandoff verfolgt dabei einen hybriden Ansatz, der nach Ansicht des Verfassers gut geeignet ist, sowohl Elemente klassischer Inline-Annotation als auch des Standoff-Verfahrens miteinander in Einklang zu bringen.

Vor diesen beiden praktisch ausgelegten Anteilen der Arbeit werden im Teil I die Grundlagen der Strukturierung linguistischer Korpora mittels texttechnologischer Methoden diskutiert. Hier werden sowohl Standardisierungsprozesse als auch die maßgeblichen Komponenten von Auszeichnungssprachen erörtert. Auch in diesem Teil werden nicht nur bereits bekannte Arbeiten wiedergegeben, vielmehr werden – neben der Verknüpfung und Neuausrichtung einiger theoretischer Ansätze – neue Erkenntnisse über die formale Basis von XML-Schemasprachen gewonnen, die in der Diskussion der Standards aufgegriffen werden, um diese zu beurteilen. Die Arbeit schließt mit Diskussion und Fazit im Teil IV ab.

1.4 Anmerkungen zu Formatierung und Zitierweise

In dieser Arbeit werden besondere Formatierungen genutzt, um bestimmte semantische Auszeichnungen zu verdeutlichen.

- Fremdsprachliche Begriffe und Namen von Organisationen werden bei erstmaliger Verwendung kursiv gesetzt (Beispiel: *W3C*).

[5] Ein Beispiel für die Institutionalisierung dieses Forschungsansatzes findet sich im bereits 1998 eingerichteten europäischen Verbundprojekt „Multilevel Annotation Tools Engineering“ (MATE). Schon hier wurden ein Markup-Framework auf Basis von XML (Dybkjær und Bernsen 2000a) und die MATE Workbench (Carletta und Isard 1999; Dybkjær und Bernsen 2000b; McKelvie *et al.* 2001) entwickelt, um vorrangig die multiple Annotation gesprochener Sprache adäquat verarbeiten zu können.

- Namen von Spezifikationen werden in Kapitälchen gesetzt (Beispiel: XML SCHEMA als der konkrete Standard vs. ein XML-Schema). Aus technischen Gründen gilt dies nicht für Überschriften (sowohl für Kapitel, Abschnitte als auch für Abbildungen und Listings).
- XML-Elemente und -Attribute werden äquidistant gesetzt (Beispiel: `segment`). Dabei wird im Allgemeinen auf Namensraum-Präfixe verzichtet.
- Attributwerte in der Instanz einer XML-Auszeichnungssprache werden kursiv gesetzt (Beispiel: *p1*).

Zu beachten ist, dass Element- und Attributnamen aus technischen Gründen auch nicht-silbenkonform getrennt werden.

Zitate erfolgen im Text nach der Verfasser-Jahr-Zitierweise (DIN 1505-3). Eine Ausnahme davon bilden Normen und andere Werke ohne eindeutig zuzuordnende Autorenschaft. Diese werden durch einen eindeutigen Bezeichner (z. B. P5 1.9.1, für die TEI GUIDELINES P5 mit der Versionsnummer 1.9.1) identifiziert.[6] Dabei gilt, dass verabschiedete Internationale Standards inkl. Jahreszahl genannt werden, deutsche und nicht in der finalen Version veröffentlichte Internationale Standards ohne Jahreszahl und letztere zusätzlich mit dem entsprechenden Statuskürzel, (vgl. ISO/DIS 24610-1, vs. ISO 24610-1:2006). Die zugehörigen Einträge im Literaturverzeichnis lauten wie folgt:

- ISO/TC 37/SC 4/WG 1 (Okt. 2005). *Language Resource Management — Feature Structures – Part 1: Feature Structure Representation*. Draft International Standard ISO/DIS 24610-1. Genf: International Organization for Standardization.

  →`http://www.tc37sc4.org/new_doc/ISO_TC_37-4_N188_Rev5_24610-1_FSR_20051020.pdf` *[Letzter Abruf: 19. 04. 2012]*

- ISO/TC 37/SC 4/WG 1 (2006). *Language Resource Management — Feature Structures – Part 1: Feature Structure Representation*. International Standard ISO 24610-1:2006. Genf: International Organization for Standardization

In den einzelnen Kapiteln wird gesondert darauf hingewiesen, welche Fassung einer Norm zur Beurteilung herangezogen wurde.[7] Im Literaturverzeichnis werden die in dieser Arbeit diskutierten Standards in einem gesonderten Teil der Bibliographie vorangestellt. Sofern vorhanden und frei zugänglich, sind URLs inkl. Datum des letzten Zugriffs bzw. DIGITAL OBJECT IDENTIFIER (DOI) angegeben.

[6] Da beim in Kapitel 7 diskutierten CORPUS ENCODING STANDARD klar zu benennende Autoren vorliegen, wird hier ebenfalls die Verfasser-Jahr-Zitierweise verwendet: Ide, Priest-Dorman *et al.* 1996.

[7] Sofern nicht für die Verortung wörtlicher Zitate benötigt, erfolgt die Referenz auf eine grundlegende Spezifikation in dieser Arbeit einmalig (beim ersten Auftreten) und wird anschließend als eingeführt betrachtet.

Teil I

Grundlagen

Angenehm bei Standards ist, dass man so viele zur Auswahl hat. Und wenn einem gar keiner gefällt, wartet man einfach auf den nächsten.

Tanenbaum (2003a, S. 263)

2 Normung und Standardisierung

Der Begriff *Standard* ist in der Umgangssprache nur sehr ungenau definiert. Als Standard gilt im Allgemeinen ein mehr oder weniger vereinheitlichter Vorgang oder ein Produkt, der oder das als allgemein verbindlich und vor allem weit verbreitet angesehen wird und dadurch als Regel oder Norm anerkannt wird. Standards im originären Sinne lassen sich in zwei Kategorien unterscheiden: De-facto- oder auch Quasi-Standards und De-jure-Standards (Norm, vgl. auch Tanenbaum 2003b, S. 90). Erstere sind ohne den Hintergedanken der Normierung entstanden (z. B. aufgrund einer besonders starken Verbreitung), letztere zeichnen sich gerade dadurch aus, dass sie im Rahmen einer gesetzlichen oder anders autorisierten Institution in den Rang einer Norm erhoben werden.[1] Dabei kann unterschieden werden zwischen De-Jure-Standards auf nationaler oder internationaler Ebene. Internationale Normen können darüber hinaus noch differenziert werden anhand der Art ihrer Normung: Aufgrund von Abkommen zwischen nationalen Regierungen (nationalen Normungsorganisationen) oder auf Basis freiwilliger, supranationaler Organisationen. Ein Beispiel für einen nationalen De-jure-Standard ist das Papierformat A4, Beispiele für De-facto-Standards sind unter anderem der IBM PC oder das von Microsoft WORD propagierte .doc-Dateiformat für Textverarbeitungsprogramme.

Weiterhin kann unterschieden werden zwischen Industriestandards, also solchen, die im Rahmen von wirtschaftlichen Prozessen eine Rolle spielen (und hierbei im Allgemeinen normiert sind, vgl. ISO 9001:2008, als international anerkannte Norm im Bereich Qualitätsmanagement), herstellerspezifischen Standards, die im Sinne eines De-facto-Standards üblicherweise durch eine gewisse Marktmacht eines einzelnen Herstellers entstehen (neben den bereits genannten Beispielen auch das Betriebssystem Microsoft WINDOWS oder das maßgeblich durch die Firma Adobe standardisierte Datenformat PDF) und offenen Standards, die durch eine Gruppe von Einzelpersonen oder

[1] Zur ersten Gruppe können je nach Sichtweise auch die sogenannten *Best Practice*-Ansätze gezählt werden, d. h., Spezifikationen, die sich in besonderer Weise als sinnvoll herausgestellt haben, ohne dass sie zwangsläufig eine große Nutzerzahl haben.

Organisationen vorangetrieben werden und die – im Gegensatz zu den beiden erstgenannten – kostenfrei bzw. unter relativ freien Lizenzen genutzt werden können (z. B. die unter dem Dach des *World Wide Web Consortiums* (*W3C*) entstandene HYPERTEXT MARKUP LANGUAGE, HTML, vgl. Raggett *et al.* 1999, die auch aufgrund ihrer freien Verfügbarkeit Grundlage für den Erfolg des World Wide Webs ist). Dabei sind die Grenzen oftmals fließend: So existiert HTML auch in Form der Norm ISO/IEC 15445:2000 und die HYPERMEDIA/TIME-BASED STRUCTURING LANGUAGE (HYTIME, ISO/IEC 10744:1997) ist nur ein Beispiel für einen frei verfügbaren ISO-Standard, der dennoch aufgrund seiner Komplexität keinerlei Anwendung gefunden hat.[2]

Im weiteren Verlauf dieser Arbeit werde sowohl Standards im engen Sinne von Normen als auch im weiteren Sinne behandelt, die die Strukturierung linguistischer Informationen ermöglichen. Dazu wird im Allgemeinen auf die Metasprache XML (vgl. Abschnitt 3.2) zurückgegriffen, die es erlaubt eigene Auszeichnungsformate (so genannte XML-Anwendungen oder Auszeichnungssprachen) zu entwickeln. Die Verwendung von XML auf breiter Front im Bereich der Annotation und strukturierten Speicherung linguistischer Korpora seit einigen Jahren ist Segen und Fluch zugleich: Zum einen erlaubt XML in einer bis dahin nicht gekannten Weise die Interoperabilität und Nachhaltigkeit der damit annotierten Daten. Zum anderen macht die nahezu grenzenlose Freiheit von XML es möglich, dass eine unüberschaubare Menge an Formaten entwickelt wurde, die genau auf den jeweiligen Einsatzzweck hin erarbeitet wurden. Zwischen diesen Formaten lässt sich zwar in vielen Fällen mittels Transformationsskripten (z. B. durch Einsatz von XSLT) übersetzen, allerdings müssen diese zunächst für jedes Paar Eingabeformat–Ausgabeformat erstellt werden, was sich erst ab einer kritischen Grenze betreffend der Anzahl der auf Grundlage eines Annotationsformats annotierten Instanzen rechnet. Insofern bleibt die im akademischen (und kommerziellen) Umfeld oft geäußerte Aussage „Unsere Daten sind nachhaltig – schließlich liegen sie in XML vor“ nur eine leere Phrase. Darüber hinaus berücksichtigt ein solcher Ansatz nicht den semantischen Gehalt des Auszeichnungsinventars: Sofern es in Annotationsformat A eine dedizierte Auszeichnung (sei es als Element oder Attribut) eines linguistischen Vorkommens gibt, die in Annotationsformat B nicht vorhanden ist, gehen Informationen verloren oder müssen – im Umkehrschluss – manuell hinzugefügt werden.

2.1 Wie entsteht ein Standard: Standardisierung in Organisationen und Normungsgremien

Simons (2007, S. 6ff.) zeigt am Beispiel der Norm ISO 639-3:2007, in welchen Schritten die Entwicklung eines Standards vonstatten geht. Die Spezifikation stellt Codes zur Referenz und Identifikation von Sprachen (aktuelle, ausgestorbene, künstliche) zur Verfügung, die aus drei Buchstaben bestehen und eine weltweit eindeutige Identifikation von Sprachen ermöglichen sollen. Ausgangspunkt für die Entwicklung dieses Standards ist die Tatsache, dass Sprachen unterschiedliche Namen haben können: Sprecher einer Sprache bezeichnen sie in ihrer Sprache oft anders als sie von Nicht-Sprechern in

[2] „Frei verfügbar“ heißt in diesem Fall, dass der Bezug des Normentexts direkt von der ISO zwar kostenpflichtig ist, eine inhaltsgleiche Fassung aber an anderer Stelle zur Verfügung steht.

anderen Sprachen bezeichnet wird. Darüber hinaus können sich Namen für Sprachen im Laufe eines längeren Zeitraums verändern. Im Zeitalter von globalen Communities, deren Einrichtung Dank der Verbreitung des Internets problemlos möglich ist, führen solche Differenzen zu erheblichen Problemen bei der Suche und Klassifizierung von Sprachdaten einer Sprache, was Simons (2007, S. 6) am Beispiel der Sprachen „Ega“, „Santa Cruz“ und „She“ plausibel darstellt. Die in Grimes (1974) vergebenen dreibuchstabigen Codes für Sprachen in der Reihenfolge internationaler Flughäfen (vgl. Gordon 2005, für eine aktuelle Version) wurde mit der freien Verfügbarkeit im Internet ein De-facto-Standard, der von verschiedenen linguistischen Gruppen genutzt wurde. 2002 wurde *SIL International* (ursprünglich *Summer Institute of Linguistics*) durch das Unterkomitee 2 des Technischen Komitees 37 der Internationalen Standardisierungsorganisation (ISO/TC 37/SC2) aufgefordert, die von Grimes (1974) gesammelten Sprachcodes zusammen mit den 400 von der ISO genutzten Kodierungen und den von der *LinguistList* entwickelten Sprachkodes für ausgestorbene und künstliche Sprachen in eine neue Spezifikation zu überführen. Das Ergebnis, ISO 639-3:2007, umfasst eine Liste von Sprachkürzeln zur einwandfreien Identifikation von knapp 7500 Sprachen. Diese schrittweise Entwicklung eines Standards, von der Konzeption im Rahmen einer eng umgrenzten Gruppe (aus dem Kreis der Industrie oder anderer interessierter Anwender, z. B. aus der Wissenschaft) über die Nutzung durch eine breitere Gemeinschaft hin zur Normierung in Form eines Internationalen Standards ist prototypisch für einige der hier besprochenen Arbeiten. Aber auch die Entwicklung einer Norm ohne direkte Vorläufer aus dem Best-Practice-Bereich ist denkbar. Im Folgenden soll auf die Standardisierung in einigen ausgewählten Institutionen verwiesen werden.

2.1.1 Normierung: Standardisierung beim Deutschen Institut für Normung und bei ISO/IEC

Das *Deutsche Institut für Normung*, kurz: *DIN e.V.*, führt im Rahmen von Arbeitsausschüssen (AA) bzw. Komitees die fachliche Arbeit der Normung durch. Dabei ist im Regelfall für eine bestimmte Normungsaufgabe genau ein Arbeitsausschuss zuständig, der diese Aufgaben sowohl in den regionalen als auch internationalen Normungsorganisationen wahrnimmt. Mehrere Arbeitsausschüsse werden zu einem Normenausschuss im *DIN* zusammengefasst, von denen es aktuell 72 gibt (vgl. DIN-Übersicht der Normenausschüsse). Der NA „trägt verantwortlich die nationale Normung auf seinem Arbeits- und Wissensgebiet und nimmt auf diesem auch die Mitarbeit bei der europäischen und internationalen Normung wahr“ (DIN-Richtlinien für Normenausschüsse, S. 3). Neue Normenausschüsse werden gegründet, sofern ein Normungsantrag vorliegt, der keinem bestehenden NA zugeordnet werden kann, bestehende NA zusammengelegt werden sollen oder ein Teilaufgabengebiet eines bestehenden Normenausschusses so umfangreich geworden ist, dass die Ausgliederung in einen neuen NA gerechtfertigt erscheint. Unter den genannten Umständen kann die Gründung vom Direktor des *DIN* dem Präsidium vorgeschlagen werden, woraufhin eine Gründungssitzung einberufen wird, in der u. a. ein Arbeitsprogramm aufgestellt wird, ein Beirat und Vorsitzende gewählt werden und gegebenenfalls Arbeitsausschüsse eingesetzt werden (DIN-Richtlinien für Normenausschüsse, S. 3f.).

NA sind eher langlebige Gebilde, wohingegen Arbeitsausschüsse relativ kurzfristig gebildet werden können. Initialzündungen für die Gründung eines AA kommen oftmals aus Kreisen einer interessierten Community – beispielsweise im Rahmen einer Fachtagung – oder auch vom *DIN* selbst, z. B., wenn zu einem internationalen Normierungsvorhaben ein nationaler Spiegelausschuss gegründet werden muss. Interessierte Mitarbeiter rekrutieren sich üblicherweise sowohl aus der Wissenschaft als auch (oftmals auch zu einem späteren Zeitpunkt) aus der Wirtschaft. Die Gründung eines AA wird vom Beirat (Lenkungsausschuss) des entsprechenden NA beschlossen, auch hier findet zunächst eine Gründungssitzung statt. Da hier die eigentliche, fachliche Arbeit durchgeführt wird, wird die Auswahl der Mitglieder durch deren fachliche Kompetenz (es sollten „die neuesten Erkenntnisse der Wissenschaft und der jeweilige Stand der Technik in die Normungsarbeit eingebracht werden“ DIN-Richtlinien für Normenausschüsse, S. 9ff.), aber auch im Sinne einer Kontinuität der Arbeit des Arbeitsausschusses bestimmt. Die Mitarbeit in den AA ist nur für akademische Mitglieder kostenfrei. Die zentrale Aufgabe eines Arbeitsausschusses ist die Erarbeitung und Verabschiedung von Normen, d. h., das Einbringen in bestehende europäische oder internationale Normungsarbeit, z. B. durch Entwicklung von Vorschlägen bzw. die Bearbeitung vorhandener Beiträge. Dazu gehört auch die Abgabe der deutschen Stellungnahme zu transnationalen Normungsvorschlägen (Norm-Entwürfen) und normativen Dokumenten, die im Rahmen der internationalen Normung auf ISO-Ebene (*International Organization for Standardization*) stattfindet. Eine solche Stellungnahme kann durch Zustimmung, Ablehnung oder Enthaltung, jeweils in Verbindung mit einem Kommentar erfolgen.

Zwei Beispiele für die Normierungsarbeit beim *DIN* und dessen internationale Einbettung sollen im Folgenden kurz skizziert werden. Als nationales Spiegelgremium zum im Rahmen des von ISO (*International Organization for Standardization*) und IEC (*International Engineering Consortium*) eingerichteten gemeinsamen technischen Komitees 1 (*Joint Technical Committee*, JTC 1), Unterkomitee 36 (Sub-Commitee 36), kurz: ISO/IEC JTC 1 SC 36 „Information Technology for Learning, Education, and Training“, wurde als Teil des Normenausschusses NA 43 (Normenausschuss Informationstechnik und Anwendungen, NIA) der Arbeitsausschuss NA 043-01-36 AA (auch NIA-01-36) „Lerntechnologien“ im Zuge der Entwicklungsbegleitenden Normung (EBN) im *DIN* eingerichtet, in dessen Rahmen an der internationalen Normungsarbeit teilgenommen wurde. Darüber hinaus wurden im Rahmen der EBN so genannte DIN SPEC, d. h., öffentlich zugängliche Dokumente, entwickelt, darunter auch PAS 1032-1 und PAS 1032-2. An diesen beiden Beispielen können gut Vorstufen der Normungsarbeit gezeigt werden. Beide als PAS (*Publicly Available Specification*)[3] veröffentlichten Dokumente wurden entwickelt in Zusammenarbeit des vom BMBF geförderten Projekts „Virtuelle Aus- und Weiterbildung Wirtschaftsinformatik“ (VAWi) und den im EBN-Referat des *DIN* angesiedelten Arbeitsgruppen „Qualität im e-Learning“ bzw. „Didaktik e-Learning“. Die Arbeitsgruppen waren Teil des oben angeführten Arbeitsausschusses und erarbeiteten Themen, die zum damaligen Zeitpunkt (2001–2005) nicht Gegenstand eines nationalen oder europäi-

[3] Den ISO/IEC Directives, Part 1, S. 31 folgend, erlangen PAS mit ihrer Publikation eine Gültigkeit von drei Jahren, die maximal erneut um die gleiche Dauer verlängert werden kann. Anschließend sollte eine PAS entweder in ein anderes normatives Dokument überführt werden oder zurückgezogen werden.

schen Normungsvorhabens waren. PAS 1032-1 wurde dann nach Veröffentlichung als Vorschlag für die internationale Normierung innerhalb von ISO/IEC JTC 1 SC 36 WG 5 (*Working Group*, Arbeitsgruppe 5) „Quality Assurance and Descriptive Framework“ eingebracht. Das Ergebnis ist die Spezifikation ISO/IEC 19796-1:2005, die ein Framework zur Beschreibung, Analyse und Implementierung von Qualitätsmanagement-Ansätzen zur Verfügung stellt.

Relevanter für die Fragestellung dieser Arbeit ist der im NA 105 „Normenausschuss Terminologie (NAT)“ angesiedelte Arbeitsausschuss NA 105-00-06 AA „Sprachressourcen“, der sich mit Fragen zur Datenhaltung (Strukturierung, Speicherung, etc.) von linguistischen Ressourcen befasst und Spiegelgremium zu ISO/TC 37/SC 4 „Language Resource Management“ ist. ISO/TC 37 „Terminology and other Language and Content Resources“ befasst sich mit der Standardisierung von Prinzipien, Methoden und Anwendungen im Bereich Terminologie und anderer Sprachressourcen im Kontext multilingualer Kommunikation und kultureller Diversität. Besagtes Unterkomitee 4 (SC 4) ist wiederum unterteilt in fünf Arbeitsgruppen (WG): „Basic descriptors and mechanisms for language resources“ (WG 1), „Annotation and representation schemes“ (WG 2), „Multilingual information representation“ (WG 3), „Lexical resources“ (WG 4) und „Workflow of language resource management“ (WG 5). Die Aktivitäten der WG2 sind seit Mitte 2011 aufgeteilt worden und teilweise in eine neue Arbeitsgruppe 6 „Linguistic Annotation“ (WG 6) überführt worden. Die ursprüngliche WG 2 entwickelt seitdem vorrangig Standards im Bereich der Semantik. In den Arbeitsgruppen werden Vorschläge für als relevant erachtete Normen entwickelt. Jede dieser später möglicherweise als Internationaler Standard veröffentlichten Arbeiten durchläuft eine Reihe von sieben Stadien (ISO/IEC Directives, Part 1, S. 20):

1. *Preliminary Stage*
2. *Proposal Stage*
3. *Preparatory Stage*
4. *Committee Stage*
5. *Enquiry Stage*
6. *Approval Stage*
7. *Publication Stage*

In der ersten Phase („Stage 00“, vgl. International harmonized stage codes) wird – z. B. durch eine externe Interessensgruppe oder aber basierend auf den Zielen eines technischen Komitees – ein so genanntes *Preliminary Work Item* (PWI, vgl. ISO/IEC Directives, Part 1) eingerichtet (typischerweise auf nationaler Ebene). Dieses wird im Rahmen der allgemeinen Komiteearbeit hin überprüft und gelangt – bei positivem Ausgang – als sogenanntes *New Work Item Proposal* (NP, neuer Arbeitsvorschlag) in die Proposal Stage („Stage 10“, vgl. International harmonized stage codes). Ziel eines NP kann ein neu zu spezifizierender Standard, die Überarbeitung oder Ergänzung einer existierenden Norm, eine technische Spezifikation oder eine PAS sein. Dieser wird

zunächst den stimmberechtigten Mitgliedern (*P-Member* – im Gegensatz zu den *O-Member*) eines technischen Komitees oder – sofern bereits vorhanden – Unterkomitees zur Abstimmung gestellt.[4] Bei Zustimmung der Mehrheit der P-Member und Bereitschaft von mindestens fünf stimmberechtigten Mitgliedern zur Mitarbeit im Rahmen eines Standardisierungsprozesses wird der NP als neues Projekt zugelassen. Zusätzlich wird ein Projektleiter bestimmt, der für die weitere organisatorische Arbeit zuständig zeichnet. Für jede Normungsaktivität kann ein nationales Mitgliedsorgan die Art seiner Mitgliedschaft (aktiv mitarbeitend oder beobachtend) festlegen. So ist es auch zulässig, nur im Rahmen eines Unterkomitees den Status von O-Member auf P-Member zu wechseln (oder umgekehrt, vgl. ISO/IEC Directives, Part 1, S 11).

Damit ist der Übergang zur Preparatory Stage gekennzeichnet. In dieser Vorbereitungsphase wird das Projekt offiziell als Teil des Arbeitsprogramms des Unterkomitees ausgezeichnet und eine Arbeitsgruppe aus Experten durch den Projektleiter einberufen, deren Aufgabe es ist, einen *Working Draft* (WD) zu erstellen, wofür eine ungefähre Zeitspanne von sechs Monaten einzuplanen ist (vgl. ISO/IEC Directives, Part 1, S 21). Dieser Arbeitsentwurf ist im Regelfall mehreren Überarbeitungen unterworfen („Stage 20.20“, vgl. International harmonized stage codes) bis die Arbeitsgruppe der Ansicht ist, dass er als technisch adäquate Lösung dem zuständigen technischen (Unter-)Komitee präsentiert werden kann. Auch in diesem Stadium ist es immer noch möglich, dass das Projekt aufgegeben und keine weitere Normierungsarbeit verfolgt wird. Ansonsten ändert sich der Status des Working Drafts in einen *Committee Draft* (CD), und die nächste Phase, Committee Stage, wird erreicht. Auf Basis des nun beim zentralen ISO-Sekretariats registrierten CD werden Kommentare von der stimmberechtigten Mitglieder des zuständigen technischen (Unter-)Komitees eingeholt, d. h., der aus einem unter größtenteils nationaler Federführung erarbeiteten Entwurf wird international bewertet. Ähnlich wie in der Arbeitsentwurfs-Phase können mehrere Versionen zirkulieren (d. h., die Version wird entsprechend an die Arbeitsgruppe zurück verwiesen), bis es zu einem positiven Abstimmungsergebnis durch die P-Member kommt. Sobald dieses erreicht ist, wird der veränderte CD als *Draft International Standard* (DIS, vorläufiger Internationaler Standard) registriert („Stage 30.99“, vgl. International harmonized stage codes). Insgesamt kann es über ein Jahr oder länger dauern, bis diese Phase abgeschlossen ist (vgl. ISO/IEC Directives, Part 1, S 21). Es folgt die Enquiry Stage (Erhebungsphase), in der der DIS (offiziell als *Enquiry Draft* bezeichnet, vgl. ISO/IEC Directives, Part 1) erstmalig durch das zentrale ISO-Sekretariat einem größeren Kreis, namentlich den nationalen ISO-Mitgliedsorganen zugänglich gemacht wird.[5] Diese haben fünf Monate Zeit, um Stellung zum Standardisierungsvorhaben zu nehmen. Stellungnahmen können in Form von Zustimmung, Ablehnung oder Enthaltung erfolgen. Zusätzlich sind Kommentare möglich, die redaktioneller oder technischer Art sein können. Stimmen zwei Drittel der stimmberechtigten P-Member des zuständigen technischen (Unter-)Komitee der Annahme des DIS zu und sind nicht mehr als 25% der Stimmen negativ, wird der Status auf den eines *Final Draft International Standard* (FDIS) geändert und die nächste Phase,

[4] P-Member nehmen aktiv an einem Normierungsunterfangen teil (*participate*), während O-Member einen Beobachterstatus wahrnehmen (*observe*, vgl. ISO/IEC Directives, Part 1, S. 11).

[5] Im Falle einer gemeinsamen Normierung durch ISO und IEC wird das zugehörige Dokument beim *International Engineering Consortium* als IEC/CDV (*Committee Draft for Vote*) bezeichnet.

Approval Stage, ist erreicht. Alternativ wird die vorliegende vorläufige Fassung der Norm zur weiteren Überarbeitung zurück an das TC/SC verweisen. Selbst jetzt ist eine Aufgabe des Standardisierungsvorhabens weiterhin möglich. Im Stadium der Approval Stage (etwa: Bewilligungsphase) wird der FDIS zur formalen Bewilligung registriert. Üblicherweise sind die zugehörigen Dokumente ab diesem Zeitpunkt aufgrund der vorangeschrittenen Reife nur noch gegen Zahlung einer Lizenzgebühr im zentralen ISO-Sekretariat erhältlich, wohingegen vorherige Versionen durchaus auch frei kursieren und Gegenstand entsprechender Veröffentlichungen beteiligter Wissenschaftler sein können. Allerdings können diese noch von der finalen Fassung der Spezifikation abweichen, so dass sich Arbeiten fälschlicherweise auf einen Standard beziehen, der in dieser Form so nicht existent ist (z. B. im Falle von LINGUISTIC ANNOTATION FRAMEWORK, vgl. Kapitel 9). Das liegt zum einen an der zu diesem Zeitpunkt immer noch nicht abgeschlossenen Entwicklung, zum anderen am unterschiedlichen Aufbau der entsprechenden Dokumenttypen: Während eine Norm einer festgelegten Struktur folgt (üblicherweise Skopus, normative Referenzen, Definition von genutzten Termini, Charakteristika der Norm, informativer Anhang), sind wissenschaftliche Arbeiten hier freier, was das Verständnis teilweise erleichtert aber auch zu Ungenauigkeiten führen kann.

Der FDIS wird den nationalen ISO-Mitgliedsorganen (auf deutscher Ebene das *DIN*) zugänglich gemacht, verbunden mit der Aufforderung ein endgültiges Verdikt abzugeben. Technische Kommentare werden in dieser Phase nicht mehr berücksichtigt, sondern erst in einer spätere Version der Norm adressiert. Auch in diesem Stadium gilt, dass eine Zwei-Drittel-Mehrheit der P-Member des zuständigen (Unter-)Komitees bei weniger als einem Viertel negativer Voten für eine Annahme als dann internationale Norm erforderlich ist. Diese wird dann, in der sechsten Phase (Publication Stage) als *International Standard* veröffentlicht. Bis zu diesem Zeitpunkt sind üblicherweise drei Jahre intensiver Arbeit vergangen (gerechnet vom Zeitpunkt der Annahme eines NP, vgl. ISO/IEC Directives, Part 1, S 21). Die Stadien Proposal Stage, Preparatory Stage und Committee Stage können übersprungen werden, um diese Zeitspanne zu verkürzen (*Fast-Track*-Verfahren, ein Beispiel dafür ist die Standardisierung von ISO/IEC 29500-1:2008, die durch Microsoft betrieben wurde, dann aber scheiterte und das reguläre Verfahren durchlief).

Generell gilt für alle internationalen Normen, dass bereits in den frühen Entwicklungsstadien der Rückbezug auf bestehende Spezifikationen und eine mögliche Zusammenarbeit mit anderen Akteuren geprüft und gegebenenfalls notwendige Schritte veranlasst werden, um unnötig Ressourcen zu investieren.

Die vollständigen Einzelheiten zur Normierung im Rahmen von ISO/IEC sind in ISO/IEC (ISO/IEC Directives, Part 1); ISO/IEC (ISO/IEC Directives, Part 2) einsehbar. Relevant für die vorliegende Betrachtung von Standards und Normen ist die folgende Quintessenz: Standards sind das Ergebnis eines langwierigen und üblicherweise mit der entsprechenden Sorgfalt verfolgten Prozesses – allerdings unterliegen die Akteure im Normierungsprozess nicht nur nationalen sondern auch wirtschaftlichen Einflüssen, weshalb das Ergebnis von Abstimmungen durchaus von politischen Interessen bestimmt werden kann.

2.1.2 Offene Standards: Standardisierung beim *W3C*

Das *World Wide Web Consortium* (*W3C*) ist ein 1994 gegründetes internationales Industriekonsortium, das vorrangig von drei Forschungseinrichtungen in den USA (*Massachusetts Institute of Technology, MIT*), Frankreich (*European Research Consortium for Informatics and Mathematics, ERCIM*) und Japan (*Keio Universität*) getragen wird und darüber hinaus Büros in 14 Ländern rund um den Erdball unterhält (unter anderem auch in Deutschland, vgl. Jacobs 2007a). Vorrangiges Ziel ist die Entwicklung von offenen Standards und Richtlinien im Bereich der Webtechnologien – als bekannte Beispiele sind hier HTML bzw. XHTML (Altheim *et al.* 2001) und natürlich XML zu nennen. Langfristige Ziele sind die Sicherstellung der Verfügbarkeit des Webs für alle Menschen und auf allen dafür geeigneten Gerätschaften (Mobiltelefone, Smartphones, Tablets, etc., vgl. Jacobs 2007c). Zu den über 320 Mitgliedern des Konsortiums zählen namenhafte Firmen wie *Microsoft*, *IBM*, *Nokia*, *Google*, *Intel* aber auch das *DFKI* und die *Fraunhofer Gesellschaft*.[6] Neben den Mitgliedern sorgt das *W3C Team* mit zur Zeit 68 Personen für die Koordination der Arbeit, trägt aber auch direkt zur Standardisierung mit eigenen Beiträgen bei. [7] Neben den beiden genannten Akteuren Mitglieder und *W3C Team* können im begrenztem Umfang auch interessierte Experten auf Einladung an der Normungsarbeit teilnehmen. Die Arbeiten des *W3C* werden intern und nach außen als Aktivitäten formuliert und laufen in verschiedenen Gruppen ab; unterschieden wird zwischen Arbeitsgruppen (*Working Group*), Interessensgruppen (*Interest Group*) und Koordinationsgruppen (*Coordination Group*). Arbeitsgruppen erarbeiten technische Spezifikationen, dazu gehörige Software, Testsuiten und Dienstleistungen und überarbeiten Beiträge anderer Gruppen. Teilnehmer einer Arbeitsgruppe sind die benannten Repräsentanten der *W3C*-Mitglieder, eingeladene externe Experten oder Mitglieder des *W3C Teams*. Interessensgruppen dagegen dienen dem Austausch von Ideen und der Zusammenführung von an einem Thema (einer Aktivität) interessierten Akteuren. Neben den Teilnehmern einer Arbeitsgruppe können hier auch interessierte nicht-eingeladene Externe durch Abonnieren einer zu diesem Thema eingerichteten Mailingliste teilnehmen. Koordinationsgruppen dienen als Bindeglied zwischen den bereits genannten Gruppen sowie Gruppierungen außerhalb des Konsortiums.

Die Standardisierung im *W3C* (vgl. Abschnitt „Introduction“ in Jacobs 2005b) verläuft in mehreren Schritten und ähnelt in vielen Punkten der bereits vom *DIN* bzw. ISO/IEC bekannten Prozedur (vgl. Abbildung 2.1 sowie Abschnitt 2.1.1).

Interesse Zunächst wird seitens eines *W3C*-Mitglieds oder im Rahmen einer öffentlichen Veranstaltung des *W3C* (Workshop) Interesse bezüglich eines speziellen Themas

[6] Eine Übersicht bietet `http://www.w3.org/Consortium/Member/List`, zuletzt abgerufen am 19.04.2012.

[7] Obwohl das *W3C* auf den Seiten des deutschen *W3C*-Büros Übersetzungen für einige seiner Dokumente anbietet, sind die Ausführungen zu den *W3C*-Prozessen und der Arbeitsweise des Konsortiums nicht vollständig übersetzt, weshalb nicht für alle englischen Originalbegriffe deutsche Entsprechungen existieren. Sofern Übersetzungen einzelner Texte vorhanden sind, ist darüber hinaus die Dateiversion älter als der englische Originaltext (vgl. Jacobs 2005a, vs. Jacobs 2007b). Aus diesem Grunde werden in diesem Abschnitt der Arbeit die englischen Termini verwendet, sofern keine adäquate offizielle Übersetzung bekannt ist.

geäußert.

Einrichtung einer Aktivität bzw. Gruppe Ist das Interesse groß genug (was z. B. durch Diskussion auf Mailinglisten des so genannten *Advisory Committees*[8] belegt werden kann), wird durch den *W3C*-Direktor ein so genanntes *Activity Proposal* (Vorschlag zur Einrichtung einer neuen Aktivität) oder eine *Working Group Charter* (Statut einer Arbeitsgruppe) entwickelt, wobei ersteres im Allgemeinen das Statut mindestens einer Arbeits-, Interessens- und/oder Koordinationsgruppe umfasst. Sofern die Mitglieder des *W3C* mit einer tiefer gehenden Arbeit an der Thematik einverstanden sind, wird die neue Aktivität offiziell begonnen und die entsprechenden Gruppen nehmen ihre Arbeit auf.

Erstellung einer Spezifikation Das Ergebnis der Arbeiten ist normalerweise die Erstellung von Spezifikationen und Richtlinien. Bis zum Status einer *W3C Recommendation*, dem *W3C*-Äquivalent einer Norm, durchläuft eine Spezifikation mehrere Reifungszustände:

1. Zunächst wird ein erster öffentlich angekündigter und zugänglicher Entwurf (*First Public Working Draft*) erstellt, der durch die Arbeitsgruppe in Hinblick auf die festgelegten Statuten überarbeitet wird. Auch wenn die Arbeit des *W3C* auf Konsens ausgelegt ist, ist zu diesem frühen Zeitpunkt keine einstimmige Zustimmung der Gruppenmitglieder notwendig, somit können auch unfertige Versionen veröffentlicht werden, um die Akzeptanz externer Interessenten zu prüfen.

2. Ist die Arbeitsgruppe mit der Überarbeitung des Entwurfs zufrieden, erfolgt das so genannte *Last Call Announcement*, das ebenfalls öffentlich erfolgen muss. Die Ankündigung soll anderen Akteuren die Möglichkeit geben, die entstehende Spezifikation auf Tauglichkeit zu überprüfen, bevor die nächste Stufe der Reifung erreicht wird.

3. Zu diesem Zeitpunkt hat die Spezifikation intern den Status einer *Candidate Recommendation* erreicht, d. h., das *W3C* geht davon aus, dass die Spezifikation technisch so weit stabil und ausgereift ist, dass es sinnvoll ist, sie beispielhaft zu implementieren, da sich sowohl die Mitglieder der Arbeitsgruppe dafür ausgesprochen haben, als auch eine breit angelegte Bewertung der Spezifikation erfolgt ist. Daher erfolgt der *Call for Implementations*. Dazu gehört auch, dass eine öffentlich einsehbare Auflistung aller Änderungen erfolgt. Liegen insgesamt zwei unabhängige Implementierungen vor und sind alle sonstigen genannten Kriterien erfüllt, erreicht das Format den Stand einer *Proposed Recommendation*.

4. Der Direktor des *W3C* ruft das *Advisory Committee* zur Überprüfung auf (*Call for Review*), wofür mindestens vier Wochen Zeit zur Verfügung stehen müssen.

[8] Das *Advisory Committee* besteht aus benannten Vertretern der *W3C*-Mitgliederorganisationen.

5. Letzte Station einer Spezifikation ist schließlich die Veröffentlichung als *W3C Recommendation*.

Ebenso wie ISO/IEC-Standards können *W3C* Recommendations ergänzt und durch neue Versionen ersetzt werden. Im Unterschied zur Normung existieren hier zwar ebenfalls klare Regeln und Strukturen, die Zeitvorgaben sind allerdings weniger strikt. Ein weiterer differenzierender Aspekt liegt im stärkeren Anwendungsbezug: Ohne vorliegende Implementierungen kann die finale Verabschiedung nicht erfolgen, was in Kontrast zu manchen ISO-Normen Nutzbarkeit und Verbreitung einer *W3C* Recommendation erleichtert und ihre prinzipiell schwächere Position gegenüber De-jure-Standards verbessert.

2.1.3 Sonstige Standardisierungsbemühungen

Neben dem *W3C* erarbeiten weitere Organisationen De-facto-Standards, die sich teilweise über Jahre hinweg bewährt haben, und die daher in dieser Arbeit behandelt werden. Zu nennen wären hier ohne Zweifel die TEI GUIDELINES, die in Kapitel 5 ausführlich behandelt werden, und deren Arbeiten Eingang in die internationale Normung gefunden haben (Kapitel 6). Die TEI GUIDELINES haben seit Dezember 2000 mit dem *TEI Consortium* einen institutionellen Rahmen gefunden, der die bis dahin in eher losen Arbeitsgruppen stattfindende Arbeit erneut aufgewertet hat (zur historischen Entwicklung sei auf Abschnitt 5.1 verwiesen).

Durch Unterstützung der Europäischen Kommission wurde bereits im Februar 1993 die *Expert Advisory Group on Language Engineering Standards* (EAGLES) eingerichtet, die Beiträge zur Standardisierung großer Sprachkorpora und damit verbundener Methoden und Werkzeuge entwickelte. Als eines der Resultate sind die EAGLES GUIDELINES (EAGLES Guidelines) zu nennen, die Aspekte zu den Themen Textkorpora, Computerlexikographie, Evaluation von NLP-Systemen, Formalismen der CL und Systeme zur Analyse und Verarbeitung gesprochener Sprache behandeln. Bedauerlicherweise sind besagte GUIDELINES nur teilweise fertig gestellt und größtenteils veraltet. Die Ergebnisse sind aber im Projekt ISLE („International Standards for Language Engineering“) aufgegangen, das 2003 beendet wurde und dessen Ressourcen weiterhin zugänglich sind.[9] So ist beispielsweise der in Kapitel 7 dieser Arbeit thematisierte CORPUS ENCODING STANDARD (CES) Teil der EAGLES GUIDELINES.

Die *Organization for the Advancement of Structured Information Standards* (OASIS) ist ein nicht-profitorientiertes Konsortium, das offene Spezifikationen u. a. im Bereich Webdienste und Electronic Publishing entwickelt, darunter DOCBOOK (vgl. Abschnitt 4.3) und DARWIN INFORMATION TYPING ARCHITECTURE (DITA, Day 2010).[10]

Das *Linguistic Data Consortium* (LDC) erarbeitet primär linguistisch motivierte Ressourcen. Darunter fallen Daten (wie Baumbanken oder Korpora, z. B. der „American

[9] Weitere Informationen hierzu unter `http://www.ilc.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm`, zuletzt abgerufen am 19.04.2012.

[10] Weitere Informationen zu OASIS sind einsehbar unter `http://www.oasis-open.org/`, zuletzt abgerufen am 19.04.2012.

National Corpus“ (ANC, vgl. Abschnitt 9.3) aber auch Werkzeuge und Spezifikationen.[11]

Die *Open Language Archive Community* (OLAC) hat in den Bereichen Metadaten, Verarbeitung und Speicherung ebenfalls eine Reihe von Best-Practice- (und damit annähernd De-facto-)Standards entwickelt (vgl. Abschnitt 12.2).[12]

Wie bereits angesprochen, sind die Grenzen zwischen internationalen Normen auf der einen Seite des Spektrums und De-facto-Standards und Best-Practices auf der anderen sind nicht immer klar definiert. Im Rahmen der vorliegenden Arbeit werden sowohl internationale Normen als auch offene Standards, die durch das *W3C* oder andere Institutionen erstellt wurden, untersucht und auf ihre Tauglichkeit für die linguistische Annotation hin überprüft. Dabei bilden fast hauptsächlich letztere den Untersuchungsgegenstand des Kapitels 3, da sie die Grundlagen zur Definition von Auszeichnungssprachen bilden, während internationale Normen im Teil II diskutiert werden.

[11] Einen Überblick zum LDC bietet die Homepage unter `http://ldc.upenn.edu/`, zuletzt abgerufen am 19.04.2012.

[12] Auch hierzu gibt die Homepage unter `http://www.language-archives.org/` nähere Auskünfte, zuletzt abgerufen am 19.04.2012.

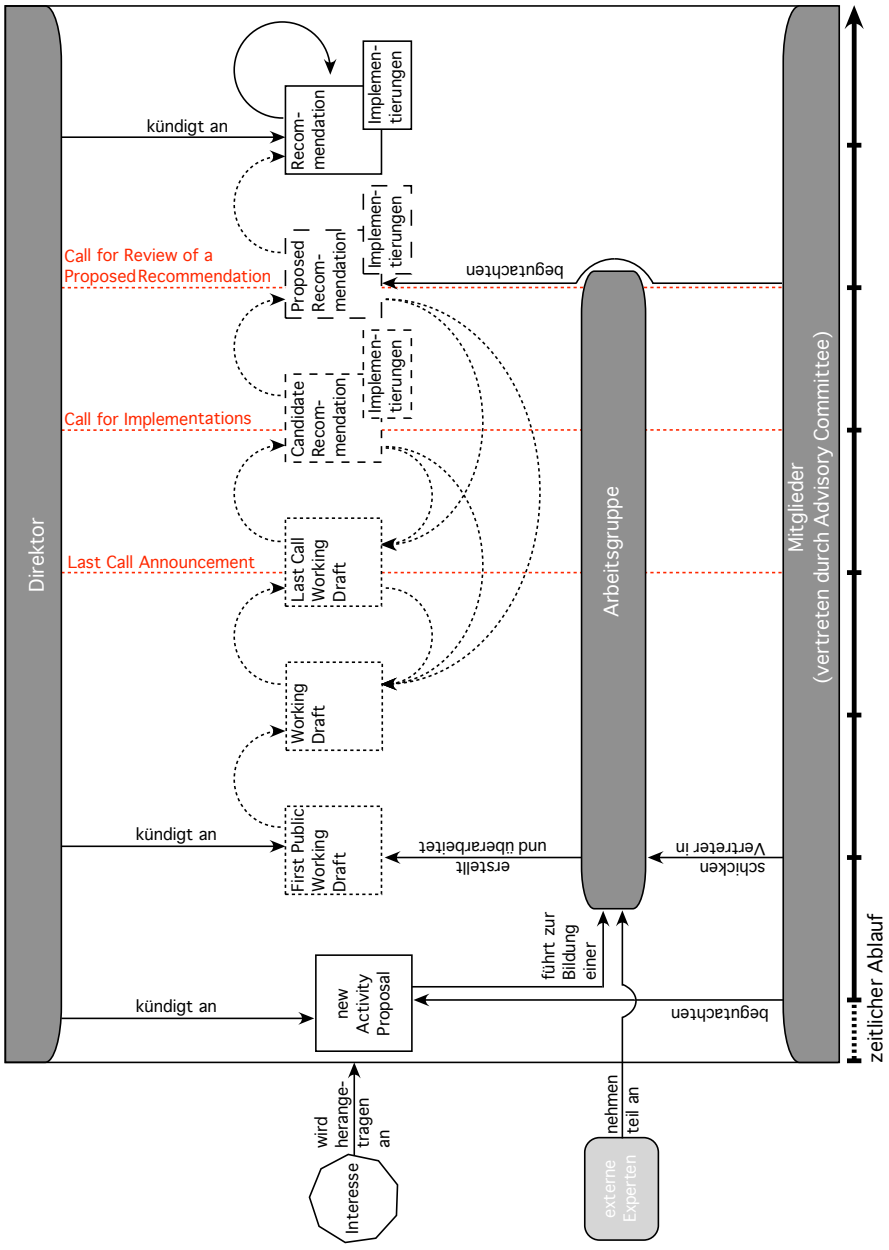


Abbildung 2.1: Eine vereinfachte Darstellung des „Recommendation Track“ des *World Wide Web Consortiums* (*W3C*)

So, wie die äußere Welt nach Aristoteles, Newton und Einstein jeweils anders gesehen wurde, wird sich Ihre Wahrnehmung eines bestimmten XML-Dokuments ändern, je nachdem, welche XML-Schemasprache Sie verwenden.

Van der Vlist (2003c, S. 370)

3

Grundlagen von Auszeichnungssprachen: Formale Modelle, Grammatikformalismen, Ebenenbegriff

Es existiert bereits eine unüberschaubare Menge an Literatur, die in die Metasprache XML und die Formulierung darauf basierender Auszeichnungssprachen einführt (darunter Behme und Mintert 1998; Harold 1999; Anderson *et al.* 2000; Lobin 2000; Evjen *et al.* 2007a, um nur einige zu nennen). Die XML-Spezifikation als solche ist nicht nur frei verfügbar online erhältlich, es existieren mit Graham und Quin (1999) und Mintert (2002) auch kommentierte (und mit letzterem auch ins Deutsche übersetzte) Fassungen. Auch das formale Modell von XML war bereits mehrfach Gegenstand von Diskussionen und es gibt eine umfassende Reihe an Arbeiten aus der akademischen Gemeinschaft, die sich mit Grammatikformalismen zur Definition von XML-Schemata auseinandersetzen (vgl. Abschnitte 3.3 und 3.4.2). Was aber fehlt, ist eine integrative Betrachtung der Grundlagen XML-basierter Auszeichnungssprachen, die sowohl das formale Modell, als auch die Grammatik sowie die Unterscheidung zwischen konzeptioneller Betrachtungsebene und Serialisierung umfasst.

Bei der Betrachtung von Auszeichnungssprachen im Rahmen dieser Arbeit kann zwischen verschiedenen Aspekten einer Auszeichnungssprache unterschieden werden: der Notation bzw. Linearisierung (Syntax), der Datenstruktur (formales Modell) und der Constraint Language, also dem Validierungsmechanismus (z. B., aber nicht ausschließlich in Form einer Grammatik). Diese drei Komponenten einer Auszeichnungssprache, erstmals aufgeführt in Sperberg-McQueen und Huitfeldt (1999a, S. 30) und in Sperberg-McQueen (2002), lassen sich als drei Säulen bezeichnen, wobei Huitfeldt und Sperberg-McQueen 2004 als weitere Metapher die drei Beine eines Stativs (*Tripod*) einführen. Überzeugend an dieser Metapher ist, dass sich ein solches Stativ nur dann sinnvoll einsetzen lässt, wenn alle drei Beine vorhanden sind – was auch für XML-basierte Auszeichnungssprachen gilt: „Both the serialization form and the data

structure stand in a natural relation, an inescapable relation, with a well-understood mechanism for validation.“ (Sperberg-McQueen 2002).

Die Notation spiegelt sich vorrangig in der *Instanz* einer Auszeichnungssprache wieder, d. h., in den jeweiligen Dateien[1], die in der entsprechenden Auszeichnungssprache annotierte Informationen enthalten.[2] Die Datenstruktur, das formale Modell, dagegen kann sowohl in der Instanz als auch in der Constraint Language betrachtet werden, wobei letztere (bzw. die zur Erstellung derselben verwendete Schemasprache) Auswirkungen auf die Datenstruktur hat, da unterschiedliche Grammatikformalismen verschiedene formale Modelle ermöglichen (vgl. Abschnitt 3.4).

Da in dieser Arbeit vorrangig XML-basierte Auszeichnungssprachen behandelt werden, ist die Syntax der Instanzen eindeutig festgelegt; bei der Diskussion von nicht auf XML basierenden Auszeichnungssprachen wird gesondert darauf hingewiesen. Die XML-Syntax ist definiert in Bray, Paoli und Sperberg-McQueen (1998); Bray, Paoli, Sperberg-McQueen, Maler und Yergeau (2008), und basiert auf der SGML-Syntax, da XML seit Hinzufügen des normativen Anhangs K, „Web SGML Adaptations“ und des informativen Anhangs L, „Additional Requirements for XML“ als Teil des *Technical Corrigendum* ISO 8879 TC2 des SGML-Standards eine echte Teilmenge von SGML ist (vgl. auch Abschnitt 3.2).[3] Davon losgelöst kann die Notation betrachtet werden, hier ist zu unterscheiden zwischen der Inline-Annotation und der Standoff-Annotation (vgl. die Abschnitte 1.2 und 3.3.2.3).

In Bezug auf das formale Modell und die verwendete Grammatik unterscheiden sich die in dieser Arbeit diskutierten Auszeichnungssprachen dagegen durchaus, weshalb auf diese Punkte im Folgenden näher eingegangen werden soll. Als Datenmodell für XML-Instanzen werden in der Literatur zwei mögliche Strukturen genannt: die des Baumes und die des Graphen (genauer: des gerichteten, azyklischen Graphen, *Directed Acyclic Graph*, DAG).[4] Sowohl Bäume als auch Graphen sind seit Jahren als Datenstrukturen in der Informatik und der Linguistik bekannt; in letzterer Disziplin werden Bäume vorrangig für die Darstellung von Parserergebnissen (Parsebäume) oder Satzstrukturen (zum Beispiel bei Phrasenstrukturgrammatiken) genutzt. Der Abschnitt 3.3 behandelt daher sowohl die beiden genannten formalen Modelle als auch eine Reihe alternativer Datenstrukturen von Auszeichnungssprachen.

Im sich daran anschließenden Abschnitt 3.4 werden die gebräuchlichsten Grammatikformalismen zur Definition einer XML-basierten Auszeichnungssprache, Document Type Definition (DTD), XML Schema Description (XSD, teilweise auch als WXS, W3C XML Schema, oder einfach XML Schema bezeichnet) und RELAX NG (RNG) vorgestellt. Dabei werden neben technischen Kriterien vorrangig Aussagen über die formale Mäch-

[1] Zu beachten ist, dass eine Instanz nicht zwangsläufig in Form einer Datei realisiert sein muss, umgangssprachlich werden allerdings beide Begriffe oftmals synonym verwendet.

[2] Dabei sind auch Annotationen ohne Inhalt, d. h., ohne annotierte Daten denkbar. Die Information wird hierbei alleine durch das Vorhandensein der entsprechenden Auszeichnung generiert.

[3] Für eine nähere Betrachtung dieser Frage wird verwiesen auf Clark (1997), sowie die zu diesem Zeitpunkt stattgefundene Diskussion in der Newsgroup comp.text.sgml, nachzulesen unter der URL `http://groups.google.com/group/comp.text.sgml/browse_thread/thread/b0c0038c6c82d11e/c3e53dee2c152a81?#c3e53dee2c152a81`, zuletzt abgerufen am 19.04.2012.

[4] Natürlich sind auch Bäume Graphen, es geht im Folgenden aber um die Unterschiede in Bezug auf die Ausdrucksstärke.

tigkeit der einzelnen Schemasprachen gemacht, wodurch sich die diskutierten Ansätze hierarchisch einordnen lassen. Ausgehend von bisherigen Arbeiten (hier vor allem Murata, D. Lee, Mani und Kawaguchi 2005) wird die neue Grammatikklasse *Unambigious Restrained Competition Grammar* eingeführt, um die strukturellen Eigenschaften der jeweiligen Formalismen genauer abbilden zu können.

3.1 Informationsstrukturierung mittels Auszeichnungssprachen

SGML- und XML-basierte Auszeichnungssprachen unterscheiden sich gegenüber anderen Formen der Datenspeicherung wie beispielsweise relationalen Datenbanken in einer zusätzlichen Form der Informationsstrukturierung. Informationen können als Merkmal-Wert-Paare gespeichert werden und nehmen dann die Form $< Merkmal >:< Wert >$ an (vgl. auch Kapitel 6). In einer relationalen Datenbank werden Merkmale und deren Werte in Spalten und Zeilen einer (oder mehrerer) Tabelle(n) strukturiert gespeichert, die Informationen sind also in zwei Dimensionen kodiert. Zusätzliche Angaben können über den Namen der Tabelle(n) und Relationen zwischen Tabellenfeldern festgehalten werden (vgl. Tabelle 3.1).[5]

Tabelle 3.1: Informationsstrukturierung in relationalen Datenbanken: Tabelle „Person“

Index	Geschlecht	Vorname	Nachname
1	Männlich	Maik	Stührenberg
2	Männlich	Max	Mustermann

In einer auf SGML oder XML basierenden Annotation können Merkmale und deren Werte über die Namen der Elemente und Attribute, deren Textinhalte bzw. Werte und zusätzlich über die hierarchische Struktur vermittelt werden (vgl. Listing 3.1).

Listing 3.1: Informationsstrukturierung in XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <personen>
3   <person geschlecht="männlich">
4    <vorname>Maik</vorname>
5    <nachname>Stührenberg</nachname>
6   </person>
7   <person geschlecht="männlich">
8    <vorname>Max</vorname>
9    <nachname>Mustermann</nachname>
10  </person>
11 </personen>
```

Während in Tabelle 3.1 die Information, dass Vor- und Nachname zu einer Entität „Person“ gehören, im Tabellennamen kodiert sind, erlaubt die inhärente Struktur von Auszeichnungssprachen, dies durch die Verschachtelung der Elemente zu speichern (zum formalen Modell vgl. Abschnitt 3.3). Die zulässige Strukturierung von Knoten in Auszeichnungssprachen wird in der jeweiligen Dokumentgrammatik festgelegt (vgl. Abschnitt 3.4). Angaben wie das Geschlecht können als Attribut realisiert werden, ebenso könnte hier ein Indexwert gespeichert werden. Diese Information ist allerdings

[5] Besagte Relationen werden tabellenextern in der Datenbank manifestiert.

vernachlässigbar (sofern sie nicht für Querverweise benötigt wird, vgl. Abschnitt 3.4.2), da die Abfolge der Elemente über die so genannte „Document Order“ festgelegt ist.[6]

3.2 Die Syntax: Von SGML zu XML[7]

Die Geschichte von XML beginnt lange vor der Veröffentlichung der ersten Fassung des W3C-Standards (Bray, Paoli und Sperberg-McQueen 1998). Ende der 1960er Jahre kamen erstmals Überlegungen zum Einsatz deskriptiven Markups („Überschrift“, „Absatz“) auf. Der als „Generic Coding“ bezeichnete Ansatz sollte als Gegenstück zum damals verbreiteten „Specific Coding“ (die Anwendung prozeduraler Auszeichnungen, die Steuerungs- oder Formatierungsanweisungen für einen bestimmten Verarbeitungsschritt enthalten), den Austausch von Informationen zwischen verschiedenen Systemen erleichtern. Den Startpunkt für diese Entwicklung hatte 1967 William Tunnicliffe von der *Graphic Communications Association* (*GCA*) gegeben: Er stellte als Vorsitzender des *GCA Composition Committee* im September 1967 auf einem Treffen des *Canadian Government Printing Office* in einer Präsentation die Idee der Trennung von Inhalt und Formatierung vor (Behme und Mintert 1998). Zum gleichen Zeitpunkt veröffentlichte Stanley Rice in New York seine Ideen für „Standardized Editorial Structures“, einen universellen Katalog von Auszeichnungselementen zur Strukturierung editorialer Inhalte. Der Direktor der *GCA*, Norman Scharpf, erkannte die Bedeutung dieser neuen Entwicklung und richtete das *GenCode Committee* ein, das ein Projekt zum Generic Coding betreute, zunächst unter dem Namen „System X“, später dann als „GenCode concept“ bekannt.[8]

Als Charles F. Goldfarb, Ed Mosher und Ray Lorie 1969 bei einem Projekt Informationen zwischen verschiedenen Computersystemen austauschen und dabei unterschiedliche Auszeichnungselemente übersetzen mussten, entwickelten sie basierend auf den genannten Vorarbeiten die Text Description Language, generische Auszeichnungselemente, die in verschiedene prozedurale Anweisungen (abhängig vom zu verarbeitenden System) übersetzt werden konnten. IBM entwickelte daraus das Projekt „Integrated Text Processing“ mit dem ersten Prototypen Integrated Textual Information Management Experiment (InTIME). In Goldfarb, Mosher *et al.* (1970) wurden erste Ergebnisse auszugsweise öffentlich präsentiert, und auch IBM erkannte das Potential einer generischen Auszeichnungssprache. 1971 entwickelte Ed Mosher die erste ausgereifte Dokumentgrammatik für die Handbücher für IBMs Telecommunications Access Method (TCAM), Jahre bevor Goldfarb das formale Konzept einer Document type description [sic!] veröffentlichte (Goldfarb 1978). Ebenfalls 1971 wurde die Text Description Language umbenannt in die Generalized Markup Language (GML),

[6] Mit dem Erfolg von XML kam die Frage auf, wie XML-kodierte Daten in relationalen Datenbanken gespeichert werden können. Wilde (2006b) untersucht einige Problemfelder, die bei der Konvertierung zwischen beiden Systemen auftreten können.

[7] Die in diesem Abschnitt beschriebenen Ereignisse basieren weitestgehend auf den Ausführungen in Goldfarb (1991b); Goldfarb (1996); SGML Users' Group (1990). Zusätzliche Quellen wurden entsprechend gekennzeichnet.

[8] Eine grafische Darstellung der historischen Entwicklung ist in Behme und Mintert (2000) im Abschnitt 2.3, „Die neue, alte Idee: Strukturorientiert schreiben“, enthalten. Zu finden auch online unter `http://www.linkwerk.com/pub/xmlidp/2000/strukturorientiert.html`, zuletzt abgerufen am 19.04.2012.

deren drei Buchstaben in der verkürzten Schreibweise nicht nur zufällig an deren Initiatoren Goldfarb, Mosher und Lorie erinnern. Der neue Name wurde erstmals in Goldfarb (1973) öffentlich vorgestellt. GML führte eine einfache Syntax ein, inklusive der nun allgegenwärtigen spitzen Klammern „<“ und „>“ zur Trennung von Auszeichnung und sonstigem Text. Auch das Konzept von Start- (<X>) und Endtag (</X>) um den Inhalt einer Auszeichnung eindeutig zu begrenzen, entstammt GML (Connolly *et al.* 1997).

1975 entwickelte Goldfarb bei IBM in San Jose GML stetig weiter. Im Rahmen einer Wirtschaftlichkeitsbetrachtung für ein Projekt zur verteilten Dokumenterstellung mittels GML entstand das Produkt DOCUMENT COMPOSITION FACILITY (DCF, intern auch SCRIPT genannt). Während bei der *GCA* weiterhin am Projekt „GenCode concept“ gearbeitet wurde, entwickelte IBM GML weiter, bis 1978 beide Projekte gemeinsam im neu gegründeten *Committee on Computer Languages for the Processing of Text* (als Teil des bestehenden *Committee on Information Processing*) des *American National Standards Institute* (*ANSI*, das amerikanische Gegenstück zum *DIN*, vgl. Abschnitt 2.1.1) die Arbeit an der Spezifikation unter dem Namen STANDARD GENERALIZED MARKUP LANGUAGE (SGML) vorantrieben. Bereits zuvor hatte es gegenseitigen Austausch über die jeweiligen Arbeiten gegeben. In Goldfarb (1978) waren erstmals zwei Dokumentgrammatiken enthalten: eine für ein allgemeines Dokument („General Document“) als Beispiel für generische Auszeichnung, und eine für den „GML Markup Guide“. Beide dienten als Vorlage im Rahmen der Standardisierungsbemühungen von SGML, zu dessen Teilnahme Goldfarb eingeladen wurde. 1980 wurde ein erster Working Draft von SGML veröffentlicht. In Goldfarb (1981) beschreibt der Autor detailliert das in GML verwendete Konzept des *Generic Markup*, der Artikel wurde in leicht veränderter Form Teil des Anhangs des Standards (Goldfarb 1991a). 1983 empfahl die *GCA* den zu diesem Zeitpunkt aktuellen sechsten *Working Draft* als GCA Standard 101-1983, daraufhin folgten 1984 drei weitere *Working-Draft*-Versionen und regelmäßige Treffen mit internationaler Beteiligung fanden im Rahmen der Arbeitsgruppe 8 des Unterkomitees 18 des JTC 1 (ISO/IEC JTC 1/SC 18/WG 8) statt. 1985 wurde eine Vorversion von SGML als *Draft Proposal for an International Standard* veröffentlicht, im Oktober desselben Jahres folgte der Draft International Standard (DIS), der vom Amt für Veröffentlichungen der Europäischen Union eingesetzt wurde (Rubinsky und Maloney 1997, S. 436). 1986, nach acht Jahren Standardisierungsarbeit, erfolgte die Veröffentlichung als Internationaler Standard ISO 8879:1986. SGML erweiterte GML um Konstrukte wie Linking, das CONCUR Merkmal (beschrieben in DeRose 1997, S. 110f., 211; sowie in Sperberg-McQueen und Huitfeldt 1999a, S. 34ff.) und vor allem das Konzept eines validierenden Parsers. Darüber hinaus etablierte die Spezifikation endgültig das Konzept des generischen (d. h., von einer bestimmten Software unabhängigen) deklarativen und kontextbezogenen Markups (im Gegensatz zur prozeduralen Auszeichnung, die oftmals Zusammenhänge zwischen einzelnen Auszeichnungselementen vermissen ließ): „The characteristics of being declarative, generic, nonproprietary, and contextual make the Standard Generalized Markup Language ‚standard‘ and ‚generalized‘“ (Maler und El Andaloussi 1995, S. 12).

Die Bedeutung der Spezifikation spiegelt sich in folgendem Zitat wider:

> Mit der Verabschiedung der *Standard Generalized Markup Language* (SGML)

> als internationalem Standard ISO 8879 im Jahre 1986 hat eine neue Zeitrechnung begonnen. SGML hat die Informationstechnologie aus den Fesseln der technischen Abhängigkeiten befreit und einen Weg eröffnet, Informationen ausschließlich auf Grundlage ihrer inneren Gesetzmäßigkeiten und ihrer Funktion zu modellieren und zu verarbeiten. (Lobin 2000).

Schätzungen zufolge waren um 1980 herum 90% der von IBM erstellten Dokumentationen in GML erstellt worden, das amerikanische Verteidigungsministerium begann in den 1980er Jahren ebenfalls GML (und später SGML) zu nutzen, die US Armee verlangte zeitweise auf SGML basierende Dokumentationen von ihren Zulieferern und Subunternehmern. Mit der 1987 begonnenen Entwicklung der TEI GUIDELINES (vgl. Kapitel 5) wurde SGML zunehmend auch für die Strukturierung linguistischer Ressourcen relevant, da nun eine deskriptive Dokumentgrammatik zur Verfügung stand, die den Anspruch vertrat, bestehende und neue Dokumente in jeglicher Sprache mittels SGML strukturiert kodieren zu können (Sperberg-McQueen und Burnard 1995, S. 21). Bereits hierbei wurde allerdings die Komplexität von SGML bemängelt und aus Gründen des vereinfachten Austauschs von Dateien auf einige optionale Merkmale verzichtet. So findet sich in Sperberg-McQueen und Burnard 1990 der folgende Hinweis:

> The SGML standard includes a wide variety of optional and basic features, some of which are the subject of controversy. This section describes the TEI's usage of these features, summarizing the reasoning which led to the SGML declaration for interchange, which is reproduced in appendix <hdref refid=zc4>.

Es folgt eine Aufstellung von SGML-Merkmalen, die in TEI-konformen Instanzen nicht erlaubt sind, darunter auch SHORTREF, SHORTTAG und OMITTAG. Folgerichtig wird in P2 eine TEI-eigene SGML-Untermenge, das TEI INTERCHANGE SUBSET beschrieben (der entsprechende Anhang, auf den im obigen Zitat verwiesen wird, der aber noch nicht Teil der TEI P1 war).[9]

1989 begann Tim Berners-Lee am CERN in Genf die Entwicklung eines Hypertext-Systems auf Basis des bereits 1980 von ihm entworfenen „Enquire“ (Berners-Lee und Fischetti 1999, S. 4, 9ff.). Als technische Basis für eine Auszeichnungssprache entschied er sich für SGML:

> There was a family of markup languages, the standard generalized markup language (SGML), already preferred by some of the world's documentation community and at the time considered the only potential document standard among the hypertext community. I developed HTML to look like a member of that family. (Berners-Lee und Fischetti 1999, S. 41).

Diese Aussage ist auch in der offiziellen HTML-Spezifikation (Raggett 1997) zu finden:

> HTML 3.2 is an SGML application conforming to International Standard ISO 8879 – Standard Generalized Markup Language. As an SGML application, the

[9] Der Anhang ist nicht Teil der Dateien, die sich unter `http://www.tei-c.org/Vault/GL/teip1.tar.gz` als P1 herunterladen lassen, zuletzt abgerufen am 19.04.2012.

> syntax of conforming HTML 3.2 documents is defined by the combination of the SGML declaration and the document type definition (DTD).

Mit dem Erfolg des World Wide Web wurde der Ruf nach weiteren standardisierten Auszeichnungssprachen laut, die sich über das Internet austauschen lassen. HTML, als Sprache für das Web bzw. zur Strukturierung von Hypertexten entwickelt, ist zu spezifisch, um beliebige Inhalte zufriedenstellend zu strukturieren. Die Spezifikation von SGML ist dagegen hochgradig komplex, was die Verarbeitung in hinreichender Zeit im Rahmen von Online-Anwendungen verhindert (Lobin 2000, S. 2). Burnard (1995) diskutiert einige der Kritikpunkte und Lösungsmöglichkeiten: Der Verzicht auf HTML kommt aufgrund der schon zum damaligen Zeitpunkt starken Verbreitung von Browsern nicht in Frage. Eine Server-seitige Verarbeitung von SGML-strukturierten Daten mit der anschließenden Transformation nach HTML erscheint genau so wenig sinnvoll (durch den zu erwartenden Workload) wie die Client-seitige (hier fehlt entsprechende Software im Sinne einer Browser-Hilfsapplikation bzw. eines SGML-Clients):

> If SGML-to-HTML servers are complex, expensive, and CPU-intensive database applications which only large corporations can afford, and hybrid clients like Panorama provide only half the functionality needed, at twice the cost, we clearly need to see a new breed of software before we can deliver on some of the promises of the world of structured documents. However sophisticated our servers, existing user agents will remain unable to take full advantage of the potential richness of the SGML documents already existing in the world, still less those which are being created, so long as they persist in regarding anything beyond HTML as outside their preserve. (Burnard 1995, Abschnitt 4, „Why not do the job properly?“).

Auch wenn Rubinsky und Maloney (1997) eine ganze Reihe von Vorschlägen zur Nutzung von SGML im Web machen, erschien die einzig sinnvolle Lösung die Entwicklung einer vereinfachten Version von SGML, vorzugsweise zusammen mit einer standardisierten Sprache zur Transformation und Formatierung derart strukturierter Inhalte. Bis zum damaligen Zeitpunkt verwendeten SGML-Browser jeweils proprietäre Stylesheet-Sprachen, die untereinander inkompatibel waren, daran änderte auch die Verabschiedung der Document Style Semantics and Specification Language (DSSSL) als Internationaler Standard ISO/IEC 10179:1996 wenig. Die adressierten Designprinzipien sind in Bray (1996a); Sperberg-McQueen und Bray (1997) zusammengefasst und werden in Goossens und Rahtz (1999) als „XML's ten commandments“ tituliert: Web-Tauglichkeit, Unterstützung für eine breite Spanne von Anwendungen, Kompatibilität zu SGML, einfache Verarbeitung, minimale Anzahl optionaler Merkmale, Lesbarkeit für Menschen, schnelle Verfügbarkeit, formale Eindeutigkeit und einfach zu erstellende Instanzen – während Knappheit in der Darstellung nur eine geringe Priorität hatte („Terseness is of minimal importance“, Bray 1996a). Diese Ziele sollten eben durch eine radikale Verschlankung von SGML erreicht werden, u. a. durch Festlegung, wann Weißraum-Zeichen (*Whitespace*) signifikant sind und wann nicht, durch Verzicht auf die sogenannte „Tag Minimization“ (d. h., Verzicht auf Start- und/oder End-Tag), die Möglichkeit, Dokumente auch ohne eine DTD als dem Standard zugehörig anzuerkennen und die zwingende Unterstützung von Unicode-Zeichensätzen. Der Verzicht auf

eine obligatorische Dokumentgrammatik war unter anderem Erfahrungen mit Texten zu verdanken, für die es oftmals nicht möglich war, eine zufrieden stellende DTD zu entwickeln:

> The requirement that SGML be valid seems in most contexts so obvious that it would never be questioned, but if document analysis of existing documents reveals violations of structure, the most appropriate model of this information in SGML terms involves invalid SGML. If the creation of invalid SGML is foreclosed for practical reasons, our most honest alternative is to enrich whatever solution we do adopt with annotations in markup that tell the truth: what we are encoding are the equivalent of parser error messages, and the fact that our document violates its basic structure in specific places is informational. (Birnbaum 1997).

Erste Ansätze einer solchen SGML-Teilmenge (*Subset*) sind übersichtlich in DeRose (1997, S. 182f.) aufgeführt, darunter BASIC SGML und MINIMAL SGML, die beide bereits in der Spezifikation selbst festgelegt wurden (Abschnitte 15.1.1 und 15.1.2, vgl. auch Goldfarb 1991b, S. 478f.), PSGML (POOR-FOLKS SGML, Sperberg-McQueen 1992), das DTDs als optional vorsieht, das bereits erwähnte TEI INTERCHANGE SUBSET (P2), SGML LITE (Bos 1995) oder die MINIMIZED GENERALIZED MARKUP LANGUAGE (MGML, Bray 1996b).

Im August 1996 begannen die Arbeiten zur Entwicklung der EXTENSIBLE MARKUP LANGUAGE (XML), die in einer extrem kurzen Zeitspanne vorangetrieben wurden:

> The basic design of XML was accomplished in eleven weeks of feverish activity under the guidance of editors Tim Bray and C. M. Sperberg-McQueen. The work started in the last few days of August, 1996, and ended with the release of the first XML draft at the SGML '96 conference in November. While it took another year to finish working out all the details, virtually every basic feature of XML as we know it today was specified in that first published draft. (Bosak 2001).

1997 stellten Sperberg-McQueen und Bray die Arbeiten auf der ACH-ALLC '97 vor (Sperberg-McQueen und Bray 1997), ein Jahr später wurde die vereinfachte Version von SGML als Spezifikation des *World Wide Web Consortium* (*W3C*) veröffentlicht (Bray, Paoli und Sperberg-McQueen 1998). Die optionalen Merkmale DATATAG, OMITTAG, RANK, LINK, CONCUR, SHORTREF, SUBDOC, und FORMAL wurden aus SGML entfernt, das Merkmal SHORTTAG, das in SGML in verschiedener Weise ein Entfallen der Start- bzw. Endtags erlaubt, wurde verboten – mit der Ausnahme, dass Attribute, für die in der Dokumentgrammatik ein Standardwert festgelegt ist, nicht in der Instanziierung des Elements stehen müssen (Sperberg-McQueen und Bray 1997). Der Konnektor AND (*Interleave*) wurde ebenfalls entfernt, genau so wie die *Inclusion Exception* und *Exclusion Exception*, die Inhaltsmodelle von Elementen stark verkomplizieren konnten.[10] Die wesentlichen Vereinfachungen fassen Connolly *et al.* (1997) übersichtlich zusammen:

[10] Eine Inklusion bzw. Exklusion gestattet das Hinzufügen bzw. Verbieten von Elementen in Teilbäumen (Inhaltsmodellen, vgl. auch Lobin 2000). Zu einer formalen Abhandlung besagter Exceptions vgl. Kilpeläinen und Wood (2001).

- Jedes Start-Tag muss durch ein passendes End-Tag geschlossen werden.
- Leere Elemente müssen ebenfalls geschlossen werden (z. B. durch die verkürzte Schreibweise <a/>).
- Alle Attributwerte müssen durch Anführungszeichen bzw. Hochkommata umschlossen werden.

Zusätzlich erlaubt die Unterscheidung von Groß- und Kleinschreibung (in Connolly *et al.* (1997) noch als Regelung, durchgängig Kleinbuchstaben zu verwenden, aufgeführt) mehr Möglichkeiten bei den Benennung von Elementen und Attributen – erfordert aber auch eine größere Sorgfalt bei der Bearbeitung von XML-Instanzen. Gerade die erste Regel wird durch ein einfaches Beispiel aus Connolly *et al.* (1997) offensichtlich:

Listing 3.2: *Tag Minimization* in SGML

```
1 <!DOCTYPE p [
2 <!ELEMENT p - O (m | #PCDATA)*>
3 <!ELEMENT m - O (#PCDATA)>
4 ]>
5 <p>Etwas Text <m> Weiterer Text
```

In SGML ist diese Form der Auszeichnung syntaktisch korrekt. Ob allerdings die Zeichenkette „Weiterer Text“ noch Teil der Auszeichnung `m` und/oder `p` ist, ist nur durch Einblick in die Dokumentgrammatik (im Beispiel eingebettet) ersichtlich, da hier festgelegt ist, ob das `m` ein leeres Element ist und ob für das Element `p` das End-Tag optional ist (gekennzeichnet durch den Großbuchstaben „O“, *omittable*, in der Deklaration, die bereits erwähnte *Tag Minimization*).[11] Sinnvolle Verhaltensregelungen im Umgang mit dem Minimization-Merkmal von SGML listet van Herwijnen (1994, S. 151ff.) auf. In XML sind die Grenzen der Auszeichnung auch ohne Dokumentgrammatik explizit, wie das folgende Listing zeigt:

Listing 3.3: XML-Instanz ohne *Tag Minimization*

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <p>Etwas Text <m> Weiterer Text </m> </p>
```

Um dem oben angeführten Zitat von Lobin ein entsprechendes für XML entgegenzusetzen, sei auf folgende Aussage von Goldfarb und Prescod (2004, S. 5) verwiesen:

> The success of the Extensible Markup Language is extraordinary: in just five years it has changed the way software is written, sold and used. All of the major software companies are enthusiastic about XML. New industry standards based upon it are released daily [...] The computer world's excitement can be summarized in two words: *information interchange*.

Während sich XML in relativ kurzer Zeit als Metasprache für die Strukturierung textueller Informationen auch in Forschung und Wissenschaft (selbst in nicht-texttechnologischen Disziplinen, vgl. die Ausführungen von Barillot und Achard 2000,

[11] Das geschilderte Problem ist insofern theoretischer Art, als dass SGML-Instanzen generell einer Dokumentgrammatik folgen, eine solche also vorhanden sein muss. Dennoch bedeutet dies für die Verarbeitung einer Instanz den zusätzlichen Schritt die Dokumentinstanz zunächst zu parsen.

zwei Biotechnologen) etabliert hat, ist rund zehn Jahre nach der Verabschiedung der Spezifikation das ursprüngliche Kernziel, die Eroberung des WWW weiterhin nicht erreicht. Im Gegenteil, seitens führender Mitglieder der Community wird XML bescheinigt, den Kampf um das Web verloren zu haben (vgl. Clark 2010). Nachdem in der XHTML-Version 1.0 (Pemberton, Altheim *et al.* 2000; Pemberton, Austin *et al.* 2002) zunächst das Element- und Attributinventar von HTML 4.01 an die technische Basis (die Syntax) von XML angepasst wurde, und mit XHTML MODULARIZATION 1.1 (Altheim *et al.* 2001; Austin *et al.* 2010) die Idee einer modularen Auszeichnungssprache für Web-Inhalte nur sehr zögerlich seitens der Browserhersteller umgesetzt wurde, wird an Stelle des vom *W3C* propagierten XHTML 2.0 (Axelsson *et al.* 2010) mit dem ursprünglich unter dem Namen WEB APPLICATIONS 1.0 entwickelten HTML5 (Hickson 2011) eine Spezifikation die zukünftige Entwicklung von Web-Anwendungen maßgeblich beeinflussen, die weder auf der SGML- noch auf der XML-Syntax beruht bzw. keine Anwendung beider Metasprachen ist.[12] XML ist allerdings als Standard zur Entwicklung von spezifischen Anwendungsformaten im Bereich der Wissenschaft (und einem unüberschaubar großem Teil der Wirtschaft) nicht mehr wegzudenken – vor allem in der Linguistik ist es die erste Wahl zur Definition von standardisierten Auszeichnungssprachen zur strukturierten Speicherung von Korpusdaten, wie die folgenden Kapitel dieser Arbeit zeigen werden. Einer der Gründe dafür ist, dass die Spezifikation nicht nur für sich allein gesehen werden darf, sondern in Kombination mit den „Satelliten"-Standards wie XPATH, XLINK, XSLT oder auch den verschiedenen Constraint Languages (vgl. Abschnitt 3.4). Bański (2001, S. 9) spricht hier zurecht von einem „XML Framework", dessen Nutzen in seiner Gesamtheit deutlich größer ist als der seiner Teile.[13] Auch Ide (2000) weist auf diesen Aspekt hin und verdeutlicht die Möglichkeiten in Bezug auf linguistische Korpora.

Daher ist es essentiell deutlich zu machen, dass der Antrieb bei der grundlegenden Entwicklung von generischen Auszeichnungssprachen die plattformunabhängige Auszeichnung von Texten (und weiteren linguistischen Daten) war und ist. Einige der im Rahmen dieser Arbeit diskutierten Spezifikationen haben darüber hinaus ihre Wurzeln in den Anfängen von SGML und sind oftmals in ihrer Ganzheit besser zu verstehen, wenn die Hintergründe der Entwicklung bekannt sind.

3.3 Ein formales Modell für XML-Auszeichnungssprachen

Anmerkung Wenn in diesem Abschnitt von XML die Rede ist, so ist damit der Inhalt der XML-Spezifikation in der Version 1.0 (Bray, Paoli und Sperberg-McQueen 1998; Bray, Paoli, Sperberg-McQueen, Maler und Yergeau 2008) bzw. in der Version 1.1 (Bray, Paoli, Sperberg-McQueen, Maler und Yergeau 2004; Bray, Paoli, Sperberg-McQueen, Maler, Yergeau und Cowan 2006) gemeint. Neben dieser existiert noch eine ganze Reihe ergänzender Spezifikationen, die die Metasprache XML um einzelne Aspekte

[12] Für HTML5 stehen zwei leicht unterschiedliche Syntaxen zur Verfügung: Die HTML-Syntax, die laut Spezifikation zu bevorzugen ist, und eine XHTML-Syntax, die zwar eine Anwendung von XML ist, aber sicherlich auch weiterhin von Browserherstellern stiefmütterlich behandelt werden wird.

[13] Dass ein solch komplexes Konstrukt aus Spezifikationen auch zu Irritationen und ernsten Problemen bei der Anwendung führen kann, hat Wilde und Glushko (2008) in Form einer umfangreichen Kritik ausgeführt.

erweitern bzw. konkretisieren oder deren Verarbeitung thematisieren. Einige davon sind Gegenstand späterer Abschnitte dieser Arbeit.

Bereits 1987 postulierten Coombs *et al.*: „Documents have a natural hierarchical structure: chapters have sections, sections have subsections, and so on, until one reaches sentences, words, and letters." (Coombs *et al.* 1987, S. 945).

Daraus wurde wenig später die OHCO-These (*Ordered Hierarchy of Content Objects*, vgl. DeRose, Durand *et al.* 1990), die besagt, dass sich Texte als eine geordnete Hierarchie von Inhaltsobjekten auffassen lassen – eine derart geordnete Hierarchie lässt sich als Baum (mit einer einzelnen Wurzel) darstellen. Da das anfängliche Ziel von Auszeichungs- und Metasprachen die Annotation von Texten war, findet sich in der Literatur recht häufig der Hinweis, dass auch Instanzen XML-basierter Auszeichnungssprachen das formale Modell des Baums nutzen (beispielsweise in Ng 2002; Fong *et al.* 2008). Das Datenmodell von XML-Instanzen wurde 2001 als sogenanntes XML Information Set (oder XML Infoset) in Cowan und Tobin (2001); Cowan und Tobin (2004) spezifiziert.[14] Auch hier wird auf den Baum verwiesen, allerdings nicht als allein mögliche Repräsentation von XML-Instanzen:

> This specification presents the information set as a modified tree for the sake of clarity and simplicity, but there is no requirement that the XML Information Set be made available through a tree structure; other types of interfaces, including (but not limited to) event-based and query-based interfaces, are also capable of providing information conforming to the XML Information Set. (Cowan und Tobin 2004, Abschnitt 1, „Introduction").

Die OHCO-Theorie wurde bereits in der Vergangenheit viel kritisiert. So benennen Renear *et al.* (1996) eine Reihe von Gegenbeispielen (u. a. multiple Hierarchien bzw. Sichtweisen) und diskutieren entsprechende Änderungen, stützen allerdings weiterhin die grundlegende Hypothese (vgl. Renear 1997). Caton (2000) argumentiert, dass die Annahme, Text sei eine geordnete, hierarchische Struktur, nur eine Sichtweise unter vielen sei. Nichtsdestotrotz repräsentiert das formale Modell des Baums noch immer die herrschende Sicht auf die Datenstruktur von XML-Instanzen, weshalb im Folgenden dieses Datenmodell kurz skizziert wird.

[14] Interessant ist, dass durch die Einführung des XML Information Set der Umgang mit XML-annotierten Daten verändert wird: Es ist nicht mehr notwendig, direkt XML-Instanzen zu verarbeiten, vielmehr genügt es Infosets zu manipulieren. Ein Beispiel für diesen Wandel stellt die Spezifikation der SOAP-Schnittstelle dar (vgl. Hadley 2002, Abschnitt „‚Big Picture' Changes"). Der unter dem Namen Simple Object Access Protocol von Microsoft entwickelte Standard zum Austausch von strukturierten Daten arbeitete ursprünglich mit XML-kodierten Daten (vgl. Box *et al.* 2000, Abschnitt 5, „SOAP Encoding"), während die im Rahmen des *W3C* entwickelte Version 1.2 das XML Infoset nutzt. So heißt es in Gudgin *et al.* (2007, Abschnitt 5, „SOAP Message Construct"): „A SOAP message is specified as an XML infoset whose comment, element, attribute, namespace and character information items are able to be serialized as XML 1.0. Note, requiring that the specified information items in SOAP message infosets be serializable as XML 1.0 does NOT require that they be serialized using XML 1.0." Ähnlich manipuliert auch XML Schema XML Infosets (vgl. Wilde 2005, S. 214). Kritikpunkte am XML Infoset und Vorschläge für eine mögliche Erweiterung dessen finden sich in Wilde (2002).

3.3.1 Der Baum als Datenmodell

Ein geordneter, gerichteter Baum $T = (V, E)$ besteht aus einer Menge von Knoten (V) und Kanten (ein geordnetes Paar von Knoten E). Ein Pfad ist „eine Folge von unterschiedlichen Knoten, in der die aufeinander folgenden Knoten durch Kanten miteinander verbunden sind“ und der dazu genutzt werden kann, die grundlegenden Eigenschaften eines Baumes zu definieren, nämlich, dass ein Baum zusammenhängend ist und keine Zyklen vorkommen (Saake und Sattler 2006, S. 330). Gibt es einen Pfad von einem Knoten k_1 hin zu einem Knoten k_2, dann ist k_1 ein *Vorfahre* von k_2, und k_2 umgekehrt ein *Nachfolger* oder *Nachfahre* von k_1, wobei auch $k_1 = k_2$ gelten darf (da jeder Knoten ein Vorfahre bzw. Nachfahre von sich selbst sein kann, vgl. Hopcroft und Ullman 1979, S. 4). Knoten, die selbst keine Nachfahren besitzen, werden als *Blätter* bezeichnet, alle anderen Knoten werden *innere Knoten* genannt. Hopcroft und Ullman (1979); Valiente (2002) folgend, gibt es genau einen Knoten, der keine Vorfahren hat, und von dem aus jeder andere Knoten erreichbar ist; der sogenannte Wurzelknoten. Jeder Knoten (mit Ausnahme des Wurzelknotens) hat *genau einen* Elternknoten und ist mit diesem durch *genau eine* Kante verbunden.

Die Menge der Kinder für jeden inneren Knoten ist vollständig geordnet, d. h., ist ein Knoten k_1 links von einem Knoten k_2, dann sind auch alle Kinder von k_1 (alle von k_1 dominierten Knoten) links von allen Kindern von k_2 (allen von k_2 dominierten Knoten). Zusätzlich sind keine kreuzenden Kanten erlaubt.[15]

Eine mögliche Interpretation der Kanten (z. B. auch in der Linguistik) ist die, dass sich Bäume größtenteils durch so genannte *Containment*-Relationen (etwa: Eingrenzung, Einschluss) definieren lassen, da Elternknoten jeweils direkt ihre(n) Kindknoten umschließen bzw. beinhalten. Abbildung 3.1 zeigt eine grafische Darstellung eines einfachen Baums, Abbildung 3.2 auf der nächsten Seite ein Beispiel für die Nutzung von Bäumen in der Linguistik zur grafischen Repräsentation einer Phrasenstruktur.

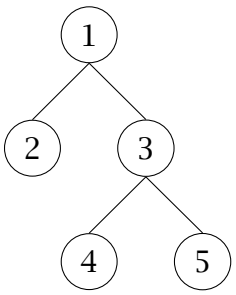


Abbildung 3.1: Ein einfacher Baum

Wie bereits angesprochen, geht man im Allgemeinen der OHCO-Theorie folgend vom Baum als Datenmodell für XML-Instanzen aus. Diese Annahme lässt sich wie folgt begründen: Betrachtet man allein die Elemente einer XML-Instanz, so ist vorgegeben,

[15] In der Linguistik werden durchaus auch kreuzende Kanten bei der Darstellung von Annotationen in Textkorpora zugelassen. Als Beispiel soll hier nur exemplarisch die TIGER-Baumbank gelten, die kreuzende Kanten zur Kennzeichnung extraponierter Relativsätze nutzt (vgl. Lezius 2002, S. 8 und Abb. 1.4 auf S. 9).

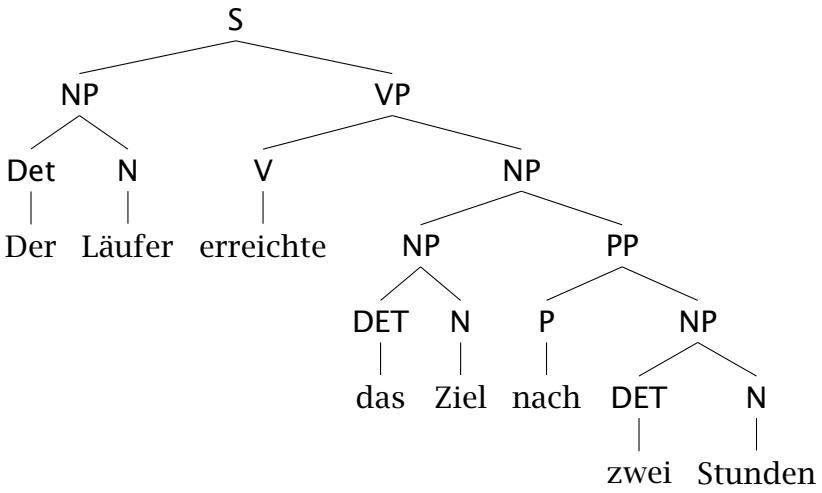


Abbildung 3.2: Nutzung von Bäumen in der Linguistik: Darstellung der Phrasen-Struktur eines Satzes

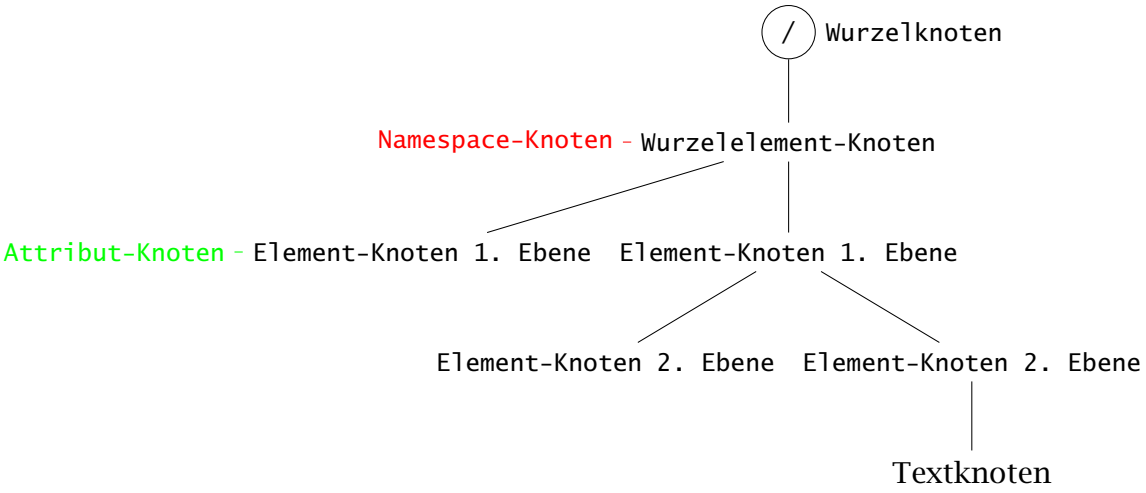


Abbildung 3.3: Das Baumdiagramm in XPath-Notation

dass das Start-Tag eines Elements *a* (<a>) *nach* dem Start-Tag eines anderen Elements *b* () und das dazu gehörige End-Tag () *vor* dem End-Tag von b () stehen muss, die beiden Elemente also ineinander verschachtelt werden können (ohne Überlappungen). In Kombination mit dem Postulat eines einzelnen Wurzelelements ergibt sich das resultierende formale Modell eines geordneten Baums. Dieses Modell wird – neben dem bereits angesprochenen XML INFORMATION SET – auch von der Abfragesprache XPATH genutzt, um die verschiedenen Knoten einer XML-Instanz (Elemente, Attribute, Namensraum-Angaben, Kommentare, Processing Instructions, Text) zu adressieren (Clark und DeRose 1999, Abschnitt 5, „Data Model“, vgl. auch Abbildung 3.3).

Ähnliches gilt für das DOCUMENT OBJECT MODEL (DOM, Apparao *et al.* 1998), ein Baum-basiertes Programmiermodell. Prinzipiell lassen sich XML-APIs (*Application Pro-*

gramming Interface, Programmierschnittstelle) in zwei Klassen unterteilen: Baum-basierte und Ereignis-basierte (*Tree based* vs. *Event based*).[16] Während erstere für gewöhnlich den vollständigen Baum im Speicher halten und damit jederzeit Zugriff auf beliebige Knoten erlauben (aber einen vergleichsweise hohen Speicherbedarf haben, vgl. Evjen *et al.* 2007d; Ammelburger 2002a), sind letztere wie das *Simple API for XML* (*SAX*, vgl. Evjen *et al.* 2007b; Ammelburger 2002b) im Allgemeinen schneller und benötigen weniger Speicher. Neben dem vom *W3C* entwickelten DOM stehen mit *JDOM*, *DOM4J* (vgl. McLaughlin und Edelson 2006; Evjen *et al.* 2007c) und *XOM* (vgl. Harold 2009) alternative Baum-basierte Programmiermodelle für die Sprache Java zur Verfügung, die aber an dieser Stelle nicht weiter behandelt werden sollen. Andere Programmiersprachen nutzen teilweise auch eigene APIs, so z. B. *Pyxie* (vgl. McGrath 2000) oder *xml.etree.ElementTree* in Python (vgl. Summerfield 2010, S. 227f.). Eine etwas veraltete Übersicht findet sich in M. C. Brown (2002).

Listing 3.4 zeigt eine einfache XML-Instanz, die sich als Baum abbilden lässt.

Listing 3.4: Einfache XML-Instanz (Baum)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <text>
3    <title>T</title>
4    <section>
5      <title>T1</title>
6      <para>P1</para>
7      <section>
8        <title>T1.1</title>
9        <para>P1.1</para>
10     </section>
11   </section>
12 </text>
```

Die grafische Repräsentation als Baum ist in Abbildung 3.4 dargestellt.

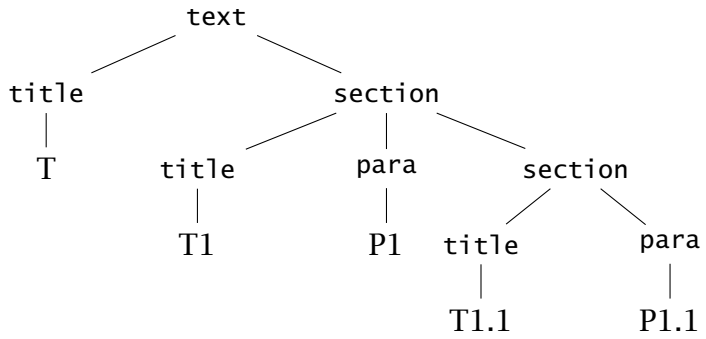


Abbildung 3.4: Baumdarstellung der XML-Instanz

Auch Klarlund, Schwentick *et al.* (2003) bezeichnen das formale Modell als *XML Tree*

[16] Die Ereignis-basierten APIs lassen sich ihrerseits in *Push* und *Pull* weiter differenzieren, wobei das von Sun propagierte StAX (*Streaming API for XML*, vgl. Harold 2003, und `http://jcp.org/aboutJava/communityprocess/final/jsr173/index.html`, zuletzt abgerufen am 19.04.2012) als Vertreter des Pull-Paradigmas angesehen wird, da hier die Anwendung den Fluss der Ereignisse (wie z. B. das Auftreten eines Start-Tags oder einer Zeichenkette) steuert („zieht“), während im klassischen Push-Ansatz der Parser eine aktivere Rolle spielt, indem er die Ereignisse der Anwendung „zuschiebt“.

und formalisieren es wie folgt (Klarlund, Schwentick *et al.* 2003, S. 9):

> **Definition 1** *An XML tree $t = (N, E, <, \lambda, \nu)$ consists of a directed tree (N, E), where N is the set of nodes, $E \subseteq N \times N$ is the set of edges, $<$ is a total order on N, $\lambda : N \to \Sigma$ is a partial labeling function, and $\nu : N \to D$ is a partial value function; moreover, the order must be depth-first: whenever x is an ancestor of y, it holds that $x < y$.*

Allerdings legen bereits Abiteboul *et al.* (2000, S. 33f.) dar, dass mit Hilfe des XML-inhärenten Verküpfungsmechanismus' (beispielsweise durch die Nutzung von Attributen des `ID`-/`IDREF`-/`IDREFS`-Token-Typs), der von einigen XML-Schemasprachen unterstützt wird, eine allgemeinere Graphenstruktur als die des Baumes repräsentiert werden kann. Ähnliche Aussagen finden sich schon für SGML in Sperberg-McQueen und Huitfeldt 1999a:

> In its simplest forms, SGML markup lends itself to a straightforward model for markup interpretation and processing: the features of the text, as the encoder understands them, are represented by SGML elements and by attributes on those elements. Since elements nest within each other and attributes apply to whole elements, an SGML document has a natural representation as a tree structure whose nodes represent elements and whose leaves represent the characters of the text. Each node is labelled with the generic identifier of its element type, and decorated with a set of attribute-value pairs representing its attributes. The legal forms of the document tree may be restricted using a document type definition (DTD), which provides a restricted form of context-free grammar; this grammar may be checked by a validating SGML parser. Links between elements, expressed in the document by ID/IDREF links, may be represented by extra arcs connecting the nodes corresponding to those elements; in this way, SGML can be used to represent not just trees but arbitrary directed graphs. (Sperberg-McQueen und Huitfeldt 1999a, S. 30)

Daher thematisiert der Abschnitt 3.3.3 das formale Modell des Graphen. Doch zunächst soll eine Unzulänglichkeit im Baummodell näher erläutert werden: Die fehlende Möglichkeit, überlappende Datenstrukturen darstellen zu können.

3.3.2 Überlappende Datenstrukturen – die Overlap-Problematik

Das Problem überlappender Informationsstrukturen in Dokumenten ist schon seit einigen Jahren Thema verschiedener Arbeiten und Forschungsbemühungen. Im englischen Sprachgebrauch wird allgemein von *Concurrent Markup* oder *Concurrent Hierarchies* gesprochen. Dekhtyar und Iacob (2005, S. 186) definieren zwei überlappende Informationsstrukturen wie folgt:

> A hierarchy is formed by a subset of the elements of the markup language used to encode the document. The elements within a hierarchy have a clear nested structure. When more than such a hierarchy is present in the markup language, the hierarchies are called concurrent.

Der englische Begriff *concurrent* wird im Deutschen für gewöhnlich mit „gleichzeitig“ oder „simultan“ übersetzt. Sinnvoller erscheint die Übersetzung „überlappend“, die genau genommen nur eine Teilmenge von Concurrent Markup bezeichnet, da natürlich auch mehrfach annotierte Daten ohne überlappende Strukturen existieren können. Da es sich aber um *den* für Forschungsfragen aktuell relevanten Aspekt handelt, und sich diese Übersetzung im deutschsprachigen Raum etabliert hat, wird sie im Folgenden genutzt. Überlappende Strukturen treten vorrangig auf, wenn Elemente aus verschiedenen Annotationsebenen oder -schichten (zur Abgrenzung des Ebenenbegriffs vgl. Abschnitt 3.5) in einer einzelnen XML-Instanz kombiniert und so die OHCO-Theorie und das Baummodell von XML verletzt werden. Diese Problematik tritt gerade bei linguistischer Annotation verstärkt auf (Sasaki und Witt 2004, S. 209):

> Durch Elemente ausgezeichnete Umgebungen dürfen sich nicht partiell überlappen. Erfolgt jedoch eine direkte Zuordnung von linguistischen Phänomenen zu der Verwendung von XML-Elementen, zeigt sich, dass derartige Überlappungen ausgesprochen häufig auftreten.

Aber auch das so genannte *Self Overlap* ist denkbar: Hier zeichnen zwei Elemente gleichen Namens (z. B. aus der gleichen Annotationsschicht) den gleichen Bereich der Primärdaten aus. Ein einfaches Beispiel aus Marinelli *et al.* (2008), ist ein Text, der von zwei Personen kommentiert wird, die dazu beide das gleiche Element nutzen (vgl. Listing 3.5).

Listing 3.5: Beispiel für Self Overlap

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <text>
3  <para>
4   Dies ist <comment>ein <comment>Text</comment></comment>.
5  </para>
6 </text>
```

Self Overlap ist mittels XML nicht serialisierbar, d. h., in der XML-Syntax nicht darstellbar, ohne die Möglichkeit der Unterscheidung zwischen einem Vorkommen von Self Overlap und korrekter Verschachtelung von Elementen zu verlieren (sofern im vorliegenden Beispiel Kommentare auch andere Kommentare beinhalten können; *Enclosure*).

Überlappende Annotationen lassen sich recht einfach festmachen, wenn man den Zeichenstrahl der Primärdaten zur Hilfe nimmt, um die Positionen der Start- und End-Tags eindeutig zu bestimmen. Aufbauend auf Durand (1999, Kapitel 3, „Operations and Conflicts in Sequences“) und Durusau und O'Donnell (2002b) führt Witt (2004) verschiedene Formen von Relationen an, die zwischen zwei Annotationsebenen bestehen können:[17]

1. *Start-tag Identity*
   Die Start-Tags der Elemente beider Annotationsebenen sind an der gleichen Position, vgl. Nummer 1 in Abbildung 3.5 auf Seite 40.

[17] In Witt, Schonefeld *et al.* (2007) finden sich diese Relationen (mit Ausnahme der letzten) unter anderen Namen ebenfalls aufgeführt.

2. *End-tag identity* (in Witt 2004, uneinheitlich als *End-point Identity* bezeichnet)
   Die End-Tags der Elemente beider Annotationsebenen sind an der gleichen Position, vgl. Nummer 2 in Abbildung 3.5 auf der nächsten Seite.

3. *Full Inclusion*
   Das Element der einen Ebene ist vollständig (sowohl Start- als auch End-Tag) eingebettet in das Element der anderen Ebene, vgl. Nummer 3 auf der nächsten Seite in Abbildung 3.5.

4. *Total Identity*
   Sowohl Start- als auch End-Tag beider Elemente sind an der gleichen Position, vgl. Nummer 4 in Abbildung 3.5 auf der nächsten Seite.

5. *Overlap*
   Das Start-Tag des einen Elements einer Annotationsebene öffnet nach dem Start-Tag des Elements der anderen Ebene und sein End-Tag schließt nach dessen End-Tag (klassische Überlappung), vgl. Nummer 5 in Abbildung 3.5 auf der nächsten Seite.

6. *Shared End/Start point*
   Das End-Tag des Elements der einen Ebene ist an der gleichen Position wie das Start-Tag des Elements der anderen Annotationsebene, vgl. Nummer 6 auf der nächsten Seite in Abbildung 3.5.

Ebenso führt Witt (2004) Meta-Relationen ein, die zwischen Elementklassen, d. h., der Menge aller Elementinstanzen zweier Annotationsebenen, vorkommen, da gerade solche interessant sind, um evtl. wechselseitige Abhängigkeiten zwischen Komponenten unterschiedlicher Annotationsebenen aufzuspüren.

Sperberg-McQueen und Huitfeldt (1999b) unterscheiden darüber hinaus bei überlappenden Strukturen zwischen „echter“ und „falscher“ Überlappung (*Spurious Overlap*). Letzter Fall liegt vor, wenn die Annotation auch ohne Überlappung repräsentiert werden kann. Das ist möglich, sofern einer der n Regionen, die von $n - 1$ überlappenden Elementen erzeugt werden, leer ist (vgl. Listings 3.6 und 3.7).

Listing 3.6: Beispiel für Spurious Overlap

```
<a><b>John likes</a> Mary</b>
<a>John likes <b></a>Mary</b>
<a>John likes <b>Mary</a></b>
```

Alle drei Beispiele (adaptiert nach Sperberg-McQueen und Huitfeldt 1999b) lassen sich in nicht überlappende Strukturen überführen, wie nachfolgend gezeigt:[18]

Listing 3.7: Auflösung des Spurious Overlap

```
<b><a>John likes</a> Mary</b>
<a>John likes</a> <b>Mary</b>
<a>John likes <b>Mary</b></a>
```

[18] Eine solche Überführung ist natürlich nur dann möglich, sofern es sich entweder um (1) wohlgeformte Instanzen handelt, d. h. eine Dokumentgrammatik nicht vorhanden ist, oder (2) die Dokumentgrammatik die wechselseitige Einbettung der Elemente a und b erlaubt.

1. Start-Tag an gleicher Position

```
<a>.................................</a>
<b>............</b>
```

2. End-Tag an gleicher Position

```
<a>................</a>
          <b>.......</b>
```

3. Vollständige Einbettung

```
<a>.................................</a>
         <b>.........</b>
```

4. Vollständige Identität

```
<a>.................................</a>
<b>.................................</b>
```

5. Überlappung

```
<a>..................</a>
            <b>.....................</b>
```

6. Start- respektive End-Tag teilen sich eine Position

```
<a>................</a>
                     <b>.............</b>
```

Abbildung 3.5: Grafische Darstellung der Relationen zwischen zwei Annotationsebenen (nach Witt 2004)

Die Annotationen sind von der ursprünglichen MECS-Notation (MULTI-ELEMENT CODE SYSTEM, Huitfeldt 1999; Sperberg-McQueen und Huitfeldt 1999a) in die XML-Notation überführt worden. MECS und sein Nachfolger TEXMECS (Huitfeldt und Sperberg-McQueen 2001) unterstützen überlappende Strukturen, stellen aber ähnlich wie die LAYERED MARKUP AND ANNOTATION LANGUAGE (LMNL, vgl. Tennison 2002; Piez 2004; Cowan, Tennison *et al.* 2006) keine Auszeichnungssprache an sich, sondern eine Metaspache (analog zu XML) mit eigener Notation dar (vgl. auch Abschnitte 3.3.3 und 3.3.4).

Auch diskontinuierliche Annotationseinheiten, die in der Literatur oftmals in einem Atemzug mit überlappenden Strukturen genannt werden, stellen für die OHCO-Theorie und das Baum-Modell von XML ein Problem dar. Ein klassisches Beispiel findet sich

in Sperberg-McQueen und Huitfeldt (2008b) in Form der ersten Sätze aus „Alice in Wonderland“ (L. Carroll 2003, S. 9): Annotiert man den Absatz wie im Listing 3.8, fällt auf, dass die Frage „and what is the use of a book without pictures or conversation?“, die sich Alice stellt, aufgrund des Einschubs „thought Alice“ in zwei separaten q-Elementen („quote“) annotiert wird, um überlappende Strukturen (und damit syntaktisch nicht korrektes XML) zu vermeiden (vgl. auch die Ausführungen in Sperberg-McQueen und Huitfeldt 2008b).

Listing 3.8: Diskontinuierliche Annotationseinheiten

```
<?xml version="1.0" encoding="UTF-8"?>
<p>
 Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do:
 once or twice she had peeped into the book her sister was reading, but it had no pictures or
 conversations in it,
 <q>and what is the use of a book,</q>
 thought Alice
 <q>without pictures or conversation?</q>
</p>
```

Nun ist es möglich, mit Hilfe von Attributen anzugeben, dass die beiden q-Elemente zusammen gehören, sie bleiben im Baum aber getrennte Knoten. Ähnliche Phänomene lassen sich auch bei Mehrwortlexemen betrachten, lexikalischen Einheiten, die aus mehreren Token bestehen. So zeigen z. B. Pianta und Bentivogli (2004), dass unterschieden werden kann zwischen *Token* (als Repräsentation der orthographischen Form), *Potential Word* (Wortstamm) und *Lexical Unit*. Der folgende Beispielsatz aus dem Italienischen (Pianta und Bentivogli 2004, S. 31) verdeutlicht das Problem:

> Coi superalcolici bisogna andarci veramente piano.
> *Übersetzt etwa: Mit Spirituosen sollte man bedächtig umgehen.*

Die lexikalische Einheit „andarci piano“ („langsam machen“), wird durch das Adverb „veramente“ in die Token „andarci“ (wiederum aufgebaut aus dem Wortstamm „andare“ und dem Enklitikon „ci“) und „piano“ getrennt. In einer Baumstruktur lässt sich das aufgrund kreuzender Kanten nicht darstellen.

Für die Annotation überlappender Strukturen in Inline-XML wurden bereits frühzeitig Lösungsvorschläge entwickelt. Neben der nahe liegenden Variante der Erstellung mehrerer Dokumente mit jeweils einer einzelnen Annotationsebene, können auch *Milestones* und *Fragments*, oder Standoff-Auszeichnungen zum Einsatz kommen. Alle genannten Lösungen werden in den TEI Guidelines (vgl. Kapitel 5) diskutiert und im Folgenden kurz skizziert.

3.3.2.1 Multiple Dokumente

Bereits die TEI Guidelines thematisieren die Verwendung multipler Instanzen: „Conceptually, the simplest method of disentangling two (or more) conflicting hierarchical views of the same information is to encode it twice (or more), each time capturing a single view.“ (P5 1.9.1, Kapitel 20, S. 620).

Größter Nachteil bei der Verwendung multipler Dokumente ist – neben der Redundanz aufgrund der mehrfachen Speicherung der Primärdaten – die aufwändige Pflege

mehrerer Dateien nach Änderung der Primärdaten, was Inkonsistenzen nach sich ziehen kann. Darüber hinaus sind die Instanzen und deren Annotationen nicht explizit miteinander verknüpft, was vor allem den Vergleich von Annotationsebenen (vgl. Abschnitt 3.5) erschwert.[19] Der verwandte Ansatz der *Twin Documents* (vgl. Marinelli *et al.* 2008) speichert nicht nur die Primärdaten redundant, sondern darüber hinaus auch noch gemeinsame Annotationselemente (*Sacred Markup* im Gegensatz zum *Profane Markup*, Annotationseinheiten, die nur einer Annotationsebene angehören). Interessanterweise argumentieren Rehm *et al.* (2010), dass die mehrfache, redundante Speicherung im Allgemeinen für eine gesteigerte Nachhaltigkeit der Daten sorgt.

3.3.2.2 Meilensteine und Fragmentierungen

Der Einsatz von *Milestones* wird bereits in frühen Fassungen der TEI GUIDELINES (P2, Abschnitt 22.3.4.3, „Milestone method“) vorgestellt. Als Milestones (in der P5 1.9.1, Kapitel 20, S. 621 auch *segment-boundary elements* bzw. *segment-boundary delimiters* genannt) werden leere Elemente bezeichnet, die jeweils den Beginn und das Ende einer Annotation definieren und die in Form von Attributen Informationen zu den ursprünglichen Elementtypen speichern. Dabei wird zwischen Elementen mit eigenem semantischem Gehalt (z. B. `lb` für einen Zeilenumbruch) und generischen Elementen (`milestone` und `anchor`) unterschieden.

Bei der Fragmentierung werden Elemente in mehrere kleinere Vorkommen (*Partial Elements*, vgl. P5 1.9.1, Kapitel 20, S. 625) aufgeteilt, die sich ohne Überlappungen in die Dokumentstruktur einbetten lassen (z. B. ein sich über mehrere Absätze erstreckender Diskurs, der in der Mitte eines Absatzes beginnt). Durch Hinzufügen des `part`-Attributs (mit den Werten *I* für *Initial*, Beginn, *M* für *Medial*, Mitte, und *F* für *Final*, Schluss) oder der Attribute `next` und `previous` kann die logische Zusammengehörigkeit der Fragmente deutlich gemacht werden – letztere Möglichkeit wird auch als *Virtual Joining* (bzw. in diesem Fall: *Chaining*) bezeichnet (vgl. P5 1.9.1, Kapitel 20, S. 626 und Abschnitt 5.5 dieser Arbeit). Self Overlap kann allerdings mit dieser Methode nicht kodiert werden. Zu guter Letzt kann auch das `join`-Element an anderer Stelle der Instanz eingesetzt werden, um Elemente, die durch einen eindeutigen Identifikator gekennzeichnet sind, virtuell zu vereinen.

3.3.2.3 Standoff-Annotation

Eine weitere alternative Repräsentation ist die Standoff-Annotation (auch *Stand-off-Annotation*).[20] Vereinfacht ausgedrückt kann jede Annotation, die von den Primärdaten separat gespeichert wird, als Standoff-Annotation gesehen werden (Larson *et al.* 2007, S. 116). Dabei ist es irrelevant, ob die Annotation vor oder nach den zu annotierenden

[19] Durch Nutzung der Zeichenpositionen der Primärdaten lässt sich eine Relation zwischen Elementen unterschiedlicher Annotationsebenen erzeugen (Witt 2002b, vgl.). Einen ähnlichen Ansatz nutzt das in Kapitel 13 diskutierte XSTANDOFF.

[20] In Ide, Priest-Dorman *et al.* (1996) ursprünglich noch als *Remote Markup* bezeichnet; McEnery *et al.* (2006a, S. 44f.) verwenden die Bezeichnung „Standalone Annotation“, Ule und Hinrichs (2004, S. 231) führen den deutschen Begriff „eigenständige Annotation“ ein.

Daten oder sogar in einer separaten Datei erfolgt (vgl. folgende Definition der TEI Workgroup on Standoff Markup 2003):

> Markup is said to be standoff, or external, when the markup data is placed outside of the text it is meant to tag: before, after, or even outside it altogether. The markup therefore points to, rather than wraps, the relevant data content.

Die früheste in der Literatur zu findende Trennung von auszuzeichnenden Daten und der konkreten Auszeichnung geht zurück auf Arbeiten im Projekt „TIPSTER Text Program", das unter der Führung der amerikanischen *Defense Advanced Research Projects Agency* (*DARPA*) in drei Phasen (TIPSTER I-III) von 1991 an Wissenschaftler aus Regierung, Industrie und Universitäten zusammenbrachte, um neue Textverarbeitungstechnologien zu entwickeln. In der Architekturbeschreibung des Systems der Phase II heißt es: „Markups may be embedded within the document or they may co-exist with it in the form of annotations, possibly containing pointers to specific locations in the document." (TIPSTER Text Phase II Architecture Concept, S. 19).

Thompson und McKelvie (1997) beschreiben drei Gründe, die eine Standoff-Annotation notwendig machen können: (1) die Primärdaten sind entweder schreibgeschützt, oder so groß, dass ein Kopieren und Ergänzen mit Annotationen nicht durchführbar wäre; (2) überlappende Annotationen können auftreten oder (3) die Primärdaten können aus rechtlichen Gründen nicht weitergegeben werden, die Auszeichnungen sollen es aber. Eine Abwandlung des ersten Punkts liegt auch immer dann vor, wenn eine Inline-Annotation aus technischen Gründen gar nicht möglich ist, z. B., weil die Primärdaten nicht in digitaler Form vorliegen (wobei hier Transkriptionen eine Annotationsbasis darstellen können) oder nicht textueller Art sind (bei der Annotation von Audio- bzw. Videodaten).[21] Auch lassen sich problemlos Annotationen von mehreren Personen gleichzeitig durchführen oder die Ausgabe verschiedener Software voneinander trennen – ein nicht unerheblicher Vorteil, da ein Großteil der linguistischen Software als Eingabeformat unannotierten Text verlangt (vgl. auch Bański 2010). Darüber hinaus hebt Bański (2001, S. 7) hervor, dass Suchoperationen einfacher sein können, da jeweils nur nach bestimmten Kriterien (und dann gegebenenfalls nur in entsprechenden Annotationen bzw. Dateien) gesucht wird. Aber auch im Sinne der von Leech (1993, S. 275) angeführten und auf S. 6 dieser Arbeit bereits dargelegten Maximen der Korpusannotation ist der Einsatz von Standoff-Notation sinnvoll.

Wynne (2005) beschreibt ebenfalls die Notwendigkeit, die Primärdaten in unveränderter Form anderen Nutzern zur Verfügung zu stellen – z. B., als Grundlage für eigene Annotationen:

> It is also likely that a well-constructed resource which is made available will have annotation added to it by other researchers. It is good practice to release a version of the corpus without annotation, however, for several reasons. Firstly, there are likely to be many users who do not wish to use the annotation, or indeed who use tools which find it difficult to process a

[21] Die gleichen Gründe finden sich auch in Ide (1998, S. 466f.).

> corpus with certain types of annotations. Secondly, the annotation process may involve changing the text in some ways, such as changing the word tokenisation, or removing certain elements. The latter can happen deliberately, or accidentally, and may not be easy to detect. It is therefore important that an original version of the corpus be available for reference purposes (Wynne 2005, S. 72).

Aus eher historischen Gründen interessant sind die Ausführungen in Nelson (1997).[22] Hier spricht sich der Erfinder des Terms „Hypertext“ und Entwickler des „Xanadu“-Projekts (als alternatives Hypertext-System zum WWW, vgl. Nelson 1982) gegen die Verwendung von Inline-Annotation („Embedded Markup“, also die Einbettung von Auszeichnungen in einer festgelegten Notation in die Primärdaten), aus. Seine Einwände beziehen sich zunächst auf die Überarbeitung annotierter Texte und den Vorgang des *Transpublishing*, ein Konzept, das aus dem „Xanadu“-Projekt herrührt, und die Referenzierung und Verknüpfung arbiträrer (Teil-)Dokumente unterschiedlichster Autoren ermöglicht – wobei auch nicht-valide Instanzen (bzw. Fragmente) entstehen können, wie Nelson argumentiert. Beide Argumente lassen sich aus heutiger Sicht entkräften: Eingebettete Auszeichnungen lassen – eine entsprechend mit der Syntax vertrauten Software vorausgesetzt – auch weiterhin eine Überarbeitung der Primärdaten zu, und das Konzept des Transpublishing konnte sich aufgrund rechtlicher Einschränkungen und mangelndem politischem Willen nicht durchsetzen – die technischen Voraussetzungen wären mit Spezifikationen wie XLink und XPointer prinzipiell vorhanden (vgl. auch DeRose 1999).[23] Der dritte in Nelson (1997) geäußerte Einwand ist jedoch nach wie vor relevant:

> I believe that embedded structure, enforcing sequence and hierarchy, limits the kinds of structure that can be expressed. The question we must ask is: What is the real structure of a thing or a document? (And does it reasonably fit the allowed spectrum of variation within the notational system?) [...] Enforcing sequence and hierarchy simply restricts the possibilities.

Damit greift Nelson die in Renear *et al.* (1996) aufgeführten Einwände gegen die OHCO-Theorie auf (wenn auch ohne direkten Bezug) – es gibt schlicht (textuelle) Strukturen, die nicht sequentiell bzw. hierarchisch aufgebaut sind und die sich mit einem eingeschränkten hierarchischen Modell nicht adäquat beschreiben lassen.

Eine Möglichkeit, Standoff zu annotieren, ist die Nutzung von Hyperlinks, die auf Segmente im Text verweisen („das erste Wort im zweiten Absatz“ – erkennbar z. B. durch Worterkennungsalgorithmen oder durch eine grundlegende Basisannotation). Ein Beispiel dafür ist die in (X)CES (vgl. Abschnitt 7.2) propagierte Nutzung eines rudimentär annotierten Primärdatums, weitere das Potsdamer Austauschformat für linguistische Annotationen (PAULA, vgl. Abschnitt 4.2), das zu diesem Zweck ein *Token File* nutzt, oder das im bereits erwähnten MATE-Projekt entwickelte gleichnamige

[22] Der Aufsatztitel ist eine Referenz an Dijkstra (1968).

[23] Es bleibt festzuhalten, dass gerade diese beiden Spezifikationen hochgradig relevant für die Umsetzung des ursprünglich von Nelson erdachten Hypertext-Gedankens sind. Umso bedauerlicher ist es, dass die Unterstützung durch Software nie wirklich ausreichend war.

Framework zur Annotation mehrfach ausgezeichneter Sprachdaten. Eine alternative Möglichkeit ist die Verwendung der Zeichenpositionen des Primärdatums zur Verortung von Annotationsgrenzen. Dieser Ansatz wird z. B. von XSTANDOFF verwendet (Teil III). Auch die in Teil II diskutierten normierten Formate wie das GRAPH ANNOTATION FORMAT (Kapitel 9) oder das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK (Kapitel 10) verwenden diesen Ansatz. Nachteilig an diesem Verfahren ist, dass Änderungen der Primärdaten zu Inkonsistenzen führen können und die Verarbeitung von Standoff-Annotationen somit spezifische Werkzeuge verlangt. So bemängeln auch McEnery *et al.* (2006a): „Even if such links can be established, they are usually prone to error. The second issue is purely practical. As far as we are aware the currently available corpus exploration tools [...] have all been designed for use with embedded annotation." (McEnery *et al.* 2006a, S. 44f.)

Zumindest der zweite Kritikpunkt (und in gewisser Hinsicht damit auch der erste) wurde durch die Entwicklung entsprechender Software adressiert (vgl. Abschnitt 16.1.1). Dennoch werden Standoff-Annotationen im Allgemeinen als robuster Mechanismus zur Darstellung heterogener Auszeichnungen angesehen: „It should be noted that [...] the standoff notation, as suggested by the TIPSTER architecture [...] is definitely a more robust option when dealing with more complex kinds of documents whose own structure may interact with the annotations." (Clément und de la Clergerie 2005, S. 91)

3.3.2.4 Weitere Ansätze

Einen generellen Überblick über die Möglichkeiten der Repräsentation überlappender Strukturen bietet Sperberg-McQueen (2007b); eine weitere sehr ausführliche Darstellung von Formaten (und Datenmodellen) findet sich in Marinelli *et al.* (2008), dabei wird unterschieden zwischen Alternativen zur XML-Serialisierung und zum XML-Datenmodell. Stührenberg und Goecke (2008) nehmen eine ähnliche Klassifizierung vor: Hier wird differenziert zwischen Prolog-basierten Formaten, XML-ähnlichen Formaten (d. h., Auszeichnungssprachen, die nicht die XML-Syntax nutzen) und XML-basierten Auszeichnungssprachen, die sich nicht auf das formale Modell des Baumes beschränken. Die erste Gruppe beruht auf Arbeiten von Sperberg-McQueen, Huitfeldt und Renear (2000); Sperberg-McQueen, Dubin *et al.* (2002). Der hier verfolgte Ansatz nutzt ein abstraktes Repräsentationsformat auf Basis einer Prolog-Faktenbasis, um Interpretationen von Auszeichnungen zu repräsentieren. Witt (2002a) erweitert dieses Konzept für die Darstellung multipler Auszeichnungen (vgl. auch Witt 2002b; Bayerl *et al.* 2003). In der Prologfaktenbasis werden die Elemente, Attribute und Textknoten der einzelnen Annotationsschichten als Prolog-Prädikate gespeichert, die folgende Informationen beinhalten: Art des Knotens (Element, Attribut, Text) als Prädikatsnamen, Namen der Annotationsschicht, Absolute Start- und Endposition des Bereichs der Primärdaten, der durch diesen Knoten annotiert wird, Position des Knotens im Baum, Name des Elements oder Attributs, Wert des Attributs (vgl. Witt 2004; Witt, Goecke, Sasaki *et al.* 2005, und Listing 13.2 auf Seite 244). In die zweite Gruppe fallen Arbeiten wie die bereits erwähnten und in den Abschnitten 3.3.3 und 3.3.4 näher erläuterten Metasprachen TEXMECS und LMNL, aber auch Ansätze wie MULTI-COLORED TREES (MCT, Jagadish *et al.* 2004) oder DELAY NODES (Le Maitre 2006). Als Vertreter der dritten Gruppe kann

XStandoff angesehen werden, dass in Teil III diskutiert wird.

Die folgenden Abschnitte befassen sich daher mit alternativen Datenmodellen, die die Abbildung überlappender Datenstrukturen unterstützen.

3.3.3 Der Graph als Datenmodell

Graphen werden definiert als eine durch eine Menge von Kanten verbundene Menge von Knoten. Ein (ungerichteter) Graph $G = (V, E)$ besteht aus einer Menge von Knoten (*Vertices*, V) und Menge von Kanten $E \subseteq V \times V$ (*Edges*, vgl. Hopcroft und Ullman 1979, S. 2). Bei ungerichteten Graphen ist die Kante (a,b) auch die Kante (b,a), d. h., die Kantenrelation ist symmetrisch. Neben den ungerichteten Graphen gibt es auch *gerichtete* (auch: *Digraph – Directed Graph*) und *gewichtete Graphen*. Gerichtete Graphen besitzen Kanten, die mit einer Angabe über deren Richtung versehen sind, d. h., jede Kante hat einen Start- und Zielknoten und es besteht nicht zwingend eine Symmetrie in der Kantenrelation (genauer spricht man von der *Adjazenzrelation*, da bei gegebener Kante (u, v) zwischen den Knoten u und v der Knoten v *adjazent* zum Knoten u ist, vgl. Cormen *et al.* 2007, S. 1083). Die gewichteten Graphen erlauben zusätzlich noch die Angabe eines Kantengewichts. Grafische Repräsentationen für drei gängige Graphentypen finden sich in der Abbildung 3.6.

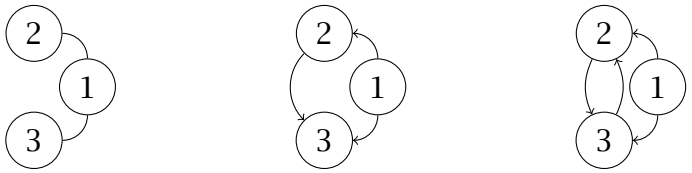


Abbildung 3.6: Ungerichteter Graph (links), gerichteter Graph ohne (Mitte) und mit Mehrfachkanten (Multigraph, rechts)

Der gerichtete, azyklische Graph (*Directed Acyclic Graph*, DAG) stellt eine weitere Unterklasse dar, die den in Abschnitt 3.3.1 bereits angesprochenen Baum umfasst. Zu beachten ist, dass zwar jeder gerichtete Baum ein DAG ist (Thulasirman und Swamy 1992, S. 118), umgekehrt aber nicht jeder DAG ein Baum ist. Azyklisch bedeutet, dass diese Graphenklasse nicht erlaubt, einen Knoten ein weiteres Mal (auf einem Rundweg) zu erreichen. DAGs sind in der Informatik sehr gut erforscht und es stehen effiziente Suchverfahren für deren Verarbeitung bereit (z. B. die Suche nach einem Knoten/einer Knotenmenge in linearer Zeit mittels *Depth-First-Search*, vgl. Bang-Jensen und Gutin 2000a; Bang-Jensen und Gutin 2000b). Ein Beispiel für gerichtete und mit einer Wurzel versehene DAGs sind Merkmalsstrukturen (vgl. Kolb 2004, S. 91), die in Kapitel 6 näher erläutert werden. Dass DAGs auch in einer XML-Notation syntaktischer Annotationen sehr nützlich sein können, zeigen Mengel und Lezius (2000).

3.3.3.1 GODDAGs

Sperberg-McQueen und Huitfeldt (2004) führen mit der Klasse der *General Ordered-Descendant Directed Acyclic Graph*s (*GODDAGs*) eine weitere Graphenklasse ein, die für

den Bereich der Auszeichnungssprachen relevant ist. Der Name rührt daher, dass es sich bei GODDAGs um Graphen handelt, deren Knoten geordnete Nachfahren haben können. Im Gegensatz zum Baum ist es in GODDAGs erlaubt, dass ein Knoten mehrere Vorfahren hat, was zur Darstellung der bereits diskutierten überlappenden Strukturen genutzt werden kann: „Overlap can be represented by graphs that are very like trees, but in which nodes may have multiple parents. Overlap is multiple parentage." (Sperberg-McQueen und Huitfeldt 2004, S. 151). Insgesamt kann zwischen *Restricted GODDAG* (kurz: *r-GODDAG*), *Generalized GODDAG*, *Clean GODDAG* (Sperberg-McQueen und Huitfeldt 2004), sowie *Normalized GODDAGs* und *Colored GODDAGs* unterschieden werden (Huitfeldt und Sperberg-McQueen 2006). Formal definiert werden GODDAGs, r-GODDAGs und Generalized GODDAGs in Sperberg-McQueen und Huitfeldt (2004) wie folgt:

Ein GODDAG ist ein gerichteter azyklischer Graph, in dem jeder Knoten entweder ein Blatt oder ein innerer Knoten (d. h. ein Knoten mit Nachfolgern bzw. Kindknoten) ist und für den gilt:

1. Jedes Blatt ist mit einem Label bestehend aus einer Zeichenkette aus null oder mehr Buchstaben versehen.

2. Jeder innere Knoten ist mit einem Label bestehend aus einem generischen Identifikator versehen.

3. Knoten werden durch gerichtete Kanten verbunden; wenn eine Kante (a, b) existiert, dominiert der Knoten a unmittelbar den Knoten b; außerdem wird b als Kind von a bezeichnet. Wenn ein gerichteter Pfad der Länge null oder größer von a zu b existiert, dominiert der Knoten a den Knoten b. Sofern der Pfad die Länge eins oder mehr hat, wird b Nachfahre von a genannt.[24]

4. Blätter sind Knoten, die nicht Ausgangspunkt einer Kante sind.

5. Innere Knoten sind alle anderen Knoten.

6. Für alle Knoten a und b gilt: Falls es Knoten $a_1, ..., a_n$ für ein $n \geq 1$ gibt und Kanten (a, a_1) und (a_i, a_{i+1}) für alle $i \leq n$ und (a_n, b) existieren, dann gibt es keine Kante (a, b).

   Das läuft darauf hinaus, dass kein Knoten einen anderen Knoten gleichzeitig direkt und indirekt dominiert.

Restricted GODDAGs sind den folgenden Einschränkungen unterworfen:

1. Die Blätter sind in einer Abfolge angeordnet. Für alle Blätter a und b gilt entweder:
   a) $a < b$ (in diesem Fall ist a b im Dokument vorangestellt), oder
   b) $a > b$ (b ist a vorangestellt), oder
   c) $a = b$ (beide Blätter sind identisch).

[24] Interessanterweise wird hier die von Hopcroft und Ullman (1979) eingeführte Definition, dass jeder Knoten auch Vorfahre und Nachfahre seiner selbst ist (also ein Pfad der Länge null existiert), negiert.

2. Jedes Blatt mit Ausnahme des ersten hat einen Vorgänger und jedes Blatt mit Ausnahme des letzten hat einen Nachfolger. Die Abfolge von Blättern wird *Frontier* (Grenze) genannt.

3. Jeder Knoten dominiert eine zusammenhängende (Teil-)Abfolge von Blättern, d. h., wenn ein Knoten n die Blätter a und b dominiert und $a < b$, dann gilt für jedes Blatt c für das $a < c < b$ gilt, n dominiert c.

4. Es gibt keine zwei Knoten, die die gleiche (Teil-)Abfolge der *Frontier* dominieren, d. h., für alle inneren Knoten a und b gibt es zumindest ein Blatt c, so dass a c dominiert und nicht b c dominiert und umgekehrt.

GODDAGs erlauben damit im Gegensatz zu den r-GODDAGs auch die Abbildung diskontinuierlicher Annotationselemente (z. B. mit Hilfe der Metasprache TEXMECS) sowie den Fall, dass zwei Elemente aus unterschiedlichen Annotationsebenen (vgl. Abschnitt 3.5) das gleiche Segment der Primärdaten dominieren. Diskontinuierliche Elemente stellen eine Kategorie der in den TEI GUIDELINES (P5 1.9.1, Kapitel 16, „Linking, Segmentation, and Alignment“) als *Virtual Elements* bezeichneten Elemente dar (die zweite wäre ein Klon eines bestehenden Elements, *Virtual Copy*).

Dagegen werden in einem Generalized GODDAG die Einschränkungen wie folgt gelockert:

1. Für jeden inneren Knoten n ist die Menge von ausgehenden Kanten geordnet. Für alle zwei Kanten (n, a) und (n, b) gilt:
   a) $(n, a) < (n, b)$
   in diesem Fall ist der Knoten a dem Knoten b unter den Kindern von n vorangestellt, oder
   b) $(n, a) > (n, b)$
   in diesem Fall ist der Knoten b dem Knoten a unter den Kindern von n vorangestellt, oder
   c) $(n, a) = (n, b)$
   in diesem Fall handelt es sich um die gleiche Kante.

2. Es gibt keine Ordnung der Blätter und die Abfolge der Blätter, die von einem inneren Knoten dominiert werden, muss nicht zusammenhängend sein (die Abwesenheit von Ordnung bedeutet, dass es keine Möglichkeit gibt, das Vorliegen einer zusammenhängenden Abfolge von Blättern zu verifizieren oder zu falsifizieren).

3. Es gibt keine Notwendigkeit, dass zwei innere Knoten unterschiedliche Mengen von Blättern dominieren.

Obwohl r-GODDAGs und GODDAGs ursprünglich als formales Modell für die Meta-Auszeichnungssprache MECS (entwickelt im Rahmen des Projekts „Markup Languages for Complex Documents“, MLCD[25], vgl. Huitfeldt 1999; Huitfeldt und Sperberg-McQueen

[25] Weitere Informationen zum Projekt unter der URL `http://mlcd.blackmesatech.com/mlcd/index.html`, zuletzt abgerufen am 19.04.2012.

2010) respektive TexMECS (Sperberg-McQueen und Huitfeldt 2008b) entworfen wurden, gelingt nicht in allen Fällen eine Abbildung zwischen r-GODDAGs und TexMECS-Linearisierung, wie Marcoux (2008a); Marcoux (2008b) zeigt. So sind z. B. alle so genannten *Clean r-GODDAGs*, d. h., solche ohne Spurious Overlap und mit nicht-leeren Zeichenketten als Blättern, in Form von OO-TexMECS-Instanzen (Overlap-only TexMECS) serialisierbar. OO-TexMECS-Instanzen können zwar überlappende Annotationen beinhalten, nicht aber diskontinuierliche oder virtuelle Elemente. Ein Beispiel für eine TexMECS-Instanz mit diskontinuierlichen Annotationseinheiten zeigt Listing 3.9 (aus Sperberg-McQueen und Huitfeldt 2008a, vgl. Listing 3.8 auf Seite 41).

Listing 3.9: TexMECS-Beispiel mit diskontinuierlichen Annotationseinheiten

```
<p id="P-6" n="Carroll, Alice I p6"|
Alice was beginning to get very tired of sitting by her sister on the bank, and of having
    nothing to do: once or twice she had peeped into the book her sister was reading, but
    it had no pictures or conversations in it,
<q|and what is the use of a book,|-q> thought Alice
<+q|without pictures or conversation?|q> |p>
```

Zu beachten ist, dass sich zwar TexMECS-Dokumente in Graphen formulieren lassen, diese Abbildung aber nicht zwingend bidirektional ist (vgl. Marcoux 2008a). Interessant in diesem Zusammenhang ist aber auch die Arbeit von Di Iorio *et al.* (2009), die mit EARMARK (Extreme Annotational RDF Markup) eine Datenstruktur auf Basis von RDF (Manola und Miller 2004; Klyne und J. J. Carroll 2004; Hayes 2004; Brickley und Guha 2004; Beckett 2004; Grant und Beckett 2004; Powers 2003) entwickelt haben, die neben der vollen Unterstützung für GODDAGs darüber hinaus auch eine erweiterte Form der GODDAGs mit anonymen inneren Knoten zulässt, die *e-GODDAGs* (*extended GODDAGs*). Die anonymen inneren Knoten können in e-GODDAGs verwendet werden, um Mehrfach-Kanten (vgl. Abbildung 3.6) zwischen zwei Knoten zu modellieren. Im Beispiel aus Di Iorio *et al.* (2009) lassen sich so mehrfach auftretende Zeilen eines Liedes elegant darstellen, ohne den textuellen Inhalt redundant speichern zu müssen. Der Text von „And I love her“ von den Beatles besteht aus mehreren Strophen und zwei unterschiedlichen Refrainzeilen (wobei die zweite in drei von vier Strophen vorkommt) und lässt sich ausschnittsweise wie folgt annotieren (übernommen aus Di Iorio *et al.* 2009):

Listing 3.10: Beatles-Beispiel aus Di Iorio *et al.* (2009)

```
<?xml version="1.0" encoding="UTF-8"?>
<body>
 <div class="stanza" title="4">
  <p>Bright are the stars that shine</p>
  <p>Dark is the sky</p>
  <p>I know this love of mine</p>
  <p>Will never die</p>
 </div>
 <div class="refrain">
  <p>And I love her</p>
 </div>
</body>
```

Abgesehen davon, dass in Di Iorio *et al.* (2009) mehrere Annotationsebenen verwendet werden (und noch zusätzliche Meta-Informationen in Form von so genannten *Fun Facts* zu bestimmten Zeitpunkten eingeblendet werden sollen), wird ein anonymer Knoten

in den e-GODDAG eingeführt, um die Refrainzeile nicht für die Strophen zwei bis vier erneut annotieren zu müssen. Dieser besitzt weder Auszeichnung noch Inhalt, sondern dient einzig der Unterscheidung der Kanten. Der Umweg über den anonymen inneren Knoten ist notwendig, um die Einschränkung zu gewährleisten, dass kein Knoten einen anderen sowohl direkt als auch indirekt dominiert. Abbildung 3.7 zeigt die aus Di Iorio *et al.* (2009) übernommene grafische Repräsentation.

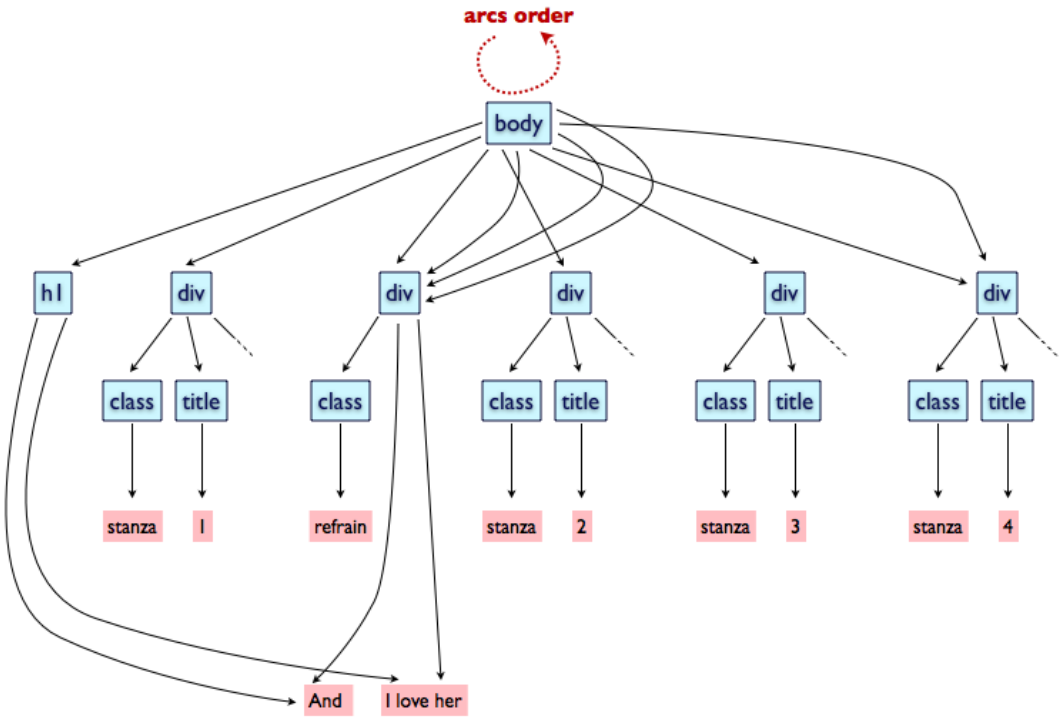


Abbildung 3.7: Grafische Darstellung des e-GODDAGs

Die mit den GODDAGs eingeführte Unterscheidung zwischen den Konzepten Dominanz (*Dominance*, die Mutter-Kindbeziehung zwischen zwei Knoten) und Einschluss (*Containment*, die Ober-/Untermenge der Blätter, die jeweils durch einen Knoten erreichbar ist, in dem der Baum nach unten traversiert wird, vgl. Abschnitt 3.3.1) kann üblicherweise in XML-basierten Auszeichnungssprachen nicht gemacht werden, da Dominanz und Einschluss durch die gleiche Mutter-Kind-Beziehung in der Baumhierarchie realisiert werden. Ein auf XML-basierendes Serialisierungsformat, das allerdings sowohl eine solche Differenzierung als auch diskontinuierliche Annotationselemente unterstützt, ist das in Teil III diskutierte XStandoff.

Inzwischen sind GODDAGs (und hier vor allem r-GODDAGs) als Datenmodell vorrangig für mehrfach annotierte Daten (inkl. möglicher überlappender Strukturen) akzeptiert

und sind auch Gegenstand verschiedener Implementierungen. So erlaubt z. B. das in Iacob und Dekhtyar (2004); Iacob und Dekhtyar (2005a); Iacob und Dekhtyar (2005b) demonstrierte Framework zur Verarbeitung von GODDAGs sowohl das Parsen von GODDAG-Datenstrukturen als auch Queries (Abfragen) darüber. Auch wenn es nicht explizit in der entsprechenden Literatur erwähnt wird, ist für den weiteren Verlauf dieser Argumentation Marinelli *et al.* (2008) folgend davon auszugehen, dass zumindest r-GODDAGs nur einen einzigen Wurzelknoten besitzen.

3.3.3.2 Weitere Graphenklassen

Eine weitere besondere Graphenklasse ist das 1999 eingeführte Modell des Annotation Graph (AG, Bird und Liberman 1999; Bird und Liberman 2001), das originär für die Verarbeitung linguistischer Annotationen entwickelt wurde. Es handelt sich hierbei um einen gelabelten DAG, der mittels einer Zuordnungsfunktion jeden einzelnen Knoten auf Punkte auf einem Zeitstrahl abbilden kann. Einfacher ausgedrückt: „Zwischen in zeitlicher Reihenfolge gegliederten Ankerpunkten werden gerichtete Graphen gespannt." (Ule und Hinrichs 2004, S. 228) Dadurch, dass die Abfolge der Ereignisse nur durch den entsprechenden Zeitpunkt, nicht aber durch die Knotenanordnung im Baum hergestellt wird, lassen sich AGs gut in XML modellieren und werden u. a. im Annotationssystem EXMARaLDA (Schmidt 2001) als Datenmodell zur Repräsentation von Mehrebenen-Annotationen eingesetzt. Zur logischen Beschreibung von Bäumen als Teil eines AG sei verwiesen auf die Ausführungen in Michaelis und Mönnich (2007). Loiseau (2007) demonstriert einen Algorithmus, der die gemeinsame Repräsentation von AG und Milestone-Elementen (und deren gegenseitige Konvertierung) erlaubt. Als negativ sehen Ide und Suderman (2007, S. 2) allerdings an, dass das Modell nicht die Verknüpfung von Annotationen untereinander erlaubt, was es ermöglichen würde, die Gesamtheit der Auszeichnungsebenen als einzigen Graph anzusehen. Wie Ule und Hinrichs (2004, S. 228) korrekt anmerken, bezahlt man generell die Ausdrucksmächtigkeit mit dem hohen Aufwand, der mit der Erstellung solcher AGs verbunden ist, wobei das frei verfügbare Annotation Graph Toolkit (AGTK) den Einsatz erleichtert.[26]

Eine dem AG nicht unähnliche (aber ihm gegenüber erweiterte) Graphenstruktur stellt das NITE Object Model (auch NXT Object Model, NOM, vgl. Evert *et al.* 2003; Carletta, Evert, Heid *et al.* 2005; Carletta, Evert, Kilgour *et al.* 2009) dar. Im aus Mitteln der Europäischen Kommission im Rahmen des HLT-Programms (Human Language Technologies) im Zeitraum vom April 2001 bis zum Juli 2003 geförderten Projekts „Natural Interactivity Tools Engineering" (NITE) wurde eine aus integrierten Werkzeugen bestehende Architektur entwickelt, die in der Lage ist, natürliche Kommunikation (sowohl zwischen menschlichen Agenten als auch Mensch-Maschine-Interaktion) in voller Bandbreite, d. h. auch über mehrere Annotationsebenen hinweg, zu annotieren und zu analysieren. Insgesamt waren acht Projektpartner am NITE-Projekt beteiligt: Das *Natural Interactive Systems Laboratory* (*NISLab*) der *University of Southern Denmark*, das *Departament de Filologia Espanyola* (*DFE*) der *Universitat Autònoma de Barcelona*, Bellaterra, das *Deut-*

[26] Weitere Informationen und Download unter `http://agtk.sourceforge.net`, zuletzt abgerufen am 19.04.2012.

sche Forschungszentrum für Künstliche Intelligenz (*DFKI*) in Saarbrücken, das *Human Communication Research Centre* (*HCRC*) der *University of Edinburgh*, das *Institut für Maschinelle Sprachverarbeitung* (*IMS*) der *Universität Stuttgart*, das *Istituto Linguistica Computazionale* (*ILC*) in Pisa, die niederländische Firma *Noldus* und das *Institut für Linguistik*, Arbeitsstelle Semiotik, der *TU Berlin* (als Subunternehmer des *IMS*).[27]

Hintergrund des NOM ist der Anspruch, vorrangig multimodale Annotationen adäquat abbilden zu können. Diese können neben der Verortung auf einer Zeitachse auch weiterhin hierarchisch strukturiert sein (in Form eines geordneten Baumes), wobei überlappende Hierarchien möglich sind. Das Objektmodell basiert daher auf einem allgemeinen Graphen. Die Knoten dieses Graphen („Elemente“) haben als Kinder entweder Text-Blätter oder aber weitere Knoten (gemischte Inhaltsmodelle sind nicht zulässig). Alle Knoten sind durch eine Zeichenkette typisiert (was in XML-Notation einem Elementnamen bzw. Elementtyp entspricht) und können darüber hinaus durch Attribut-Wert-Paare, Timing-Informationen und Zeigern (entweder auf Knoten im gleichen Graphen oder auf externe Daten) ergänzt werden. Kindknoten werden durch ihre Elternknoten dominiert und durch eine geordnete Liste repräsentiert. Je nach Knotentyp sind unterschiedliche Attribute möglich, die optionalen Timing-Informationen werden über festgelegte Start- und End-Attribute mit numerischen Werten (als Offset ausgehend vom Startpunkt des Aufnahmesignals) dargestellt. Zeiger sind als Liste von „role and filler pairs“ (Carletta, Evert, Kilgour *et al.* 2009, S. 12) kodiert, wobei 1:1- und 1:*n*-Beziehungen zwischen Rolle und Zielknoten bzw. -knotenliste (bei internen Zeigern) respektive Rolle und Datum (bei externen Zeigern) möglich sind. Strukturell interessant ist die Einschränkung, dass Pfade zwischen Eltern- und Kindknoten azyklisch sein müssen, um die Dominanzbeziehung ausdrücken zu können, und es nur einen Pfad zwischen zwei (beliebigen) Elementknoten geben darf, was im Ergebnis mehrwurzelige Bäume erlaubt:

> Firstly, the parent-child relationships in this graph must be acyclic, so that its transitive closure can be interpreted as a dominance relation. Secondly, there must not be more than one path between any two elements. Because of these constraints, the parent-child graph (which, unlike a tree, allows children to have multiple parents) decomposes into a collection of intersecting tree-like structures, called hierarchies. Each hierarchy has its own structural ordering (similar to an ordered tree), but these orderings must be consistent where hierarchies intersect. (Carletta, Evert, Kilgour *et al.* 2009, S. 12).

Ebenso werden die Timing-Informationen genutzt, um die sequentielle Abfolge der Knoten aufrecht zu halten. Der Startzeitpunkt eines Elementknotens muss kleiner bzw. gleich seinem Endzeitpunkt sein; das Zeitintervall, das durch die Timing-Informationen der Kindknoten vorgegeben ist, muss im Zeitintervall der dominierenden Elternknoten

[27] Die im Projekt erarbeiteten drei Toolsets NWB (NITE WorkBench for Windows), NXT (NITE XML Toolkit) und The Observer wurden von verschiedenen Projektpartnern erstellt und sind – mit Ausnahme des von der niederländischen Firma *Noldus* kommerziell vermarkteten Programms The Observer – frei erhältlich. Weitere Informationen finden sich unter `http://groups.inf.ed.ac.uk/nxt/`, zuletzt abgerufen am 19.04.2012.

enthalten sein. Die Relation, die durch Zeiger zwischen Knoten ausgedrückt wird, unterliegt dagegen keiner hierarchischen oder zeitlichen Beschränkung.

3.3.3.3 XML-Repräsentation von Graphen

Wie bereits im Abschnitt 3.3.1 angesprochen, wird in Abiteboul *et al.* (2000, S. 33f.) darauf verwiesen, dass mit Hilfe des XML-eigenen Verknüpfungsmechanismus' eine Graphenstruktur repräsentiert werden kann. Zu diesem Zweck wird die Instanz aus Listing 3.4 auf Seite 36 um zwei Einheiten der Informationsstrukturierung erweitert: Ein dem Element `para` zugehöriges `ID`-Token-Typ-Attribut (der Einfachheit halber mit Namen `id`) und ein Element `xref`, das als optionales leeres Element innerhalb von `para`-Elementen auftreten kann und Träger eines `IDREF`-Token-Typ-Attribut namens `idref` ist (Zeile 10 im Listing 3.11).

Listing 3.11: Einfache XML-Instanz (Graph)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <text xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="text.xsd" author="maik">
4    <title>T</title>
5    <section>
6      <title>T1</title>
7      <para>P1</para>
8      <section>
9        <title>T1.1</title>
10       <para id="p1">P1.1 <xref idref="p1"/></para>
11     </section>
12   </section>
13 </text>
```

Diese Instanz lässt sich entsprechend nur noch als Graph repräsentieren (vgl. Abbildung 3.8).

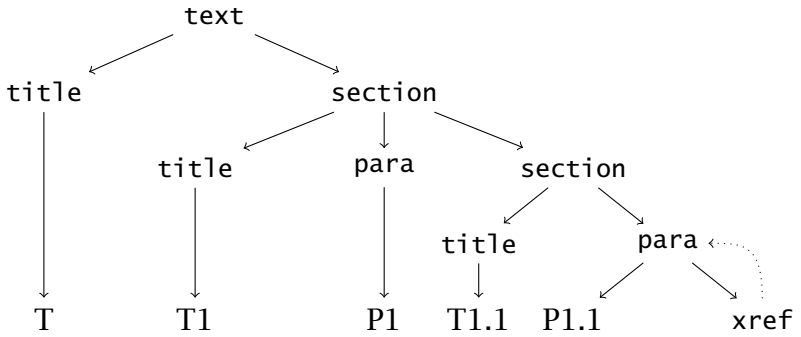


Abbildung 3.8: Graphen-Darstellung der XML-Instanz inkl. zyklischem Pfad

Ganz offensichtlich kann durch die Einbeziehung neuer `ID`/`IDREF`-Kanten (der Einfachheit halber Link-Kanten genannt) sogar das Postulat einer eindeutigen Wurzel umgangen werden, wie die Darstellung in Abbildung 3.9 auf der nächsten Seite zeigt.[28]

[28] Das Beispiel setzt eine entsprechend modifizierte Instanz voraus, d. h. konkret, das Hinzufügen eines weiteren `ID`-Token-Typ-Attributs unterhalb des `para`-Elements und die Änderung des Token-Typs des `idref`-Attributs in `IDREFS`.

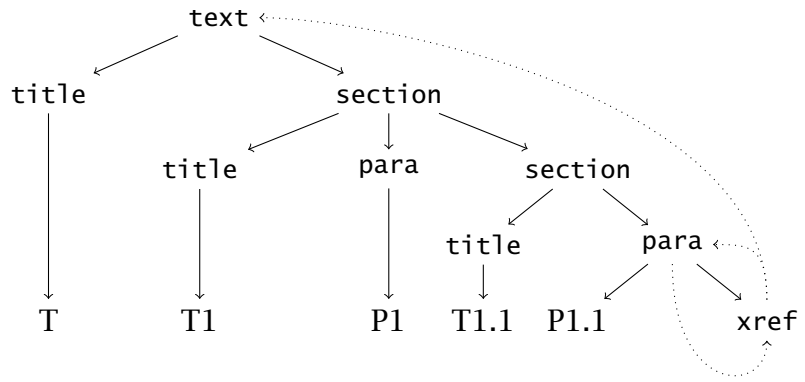


Abbildung 3.9: Graphen-Darstellung der XML-Instanz inkl. mehrerer zyklischer Pfade

Da eine minimale XML-Instanz aus einem leeren Element besteht, lässt sich die Verwendung zyklischer Pfade sogar noch weiter führen. Das Listing 3.12 zeigt eine solche minimale valide XML-Instanz (der Einfachheit halber mit eingebetteter DTD), die über zwei Attribute `id` und `idref` eine Selbstreferenz erlaubt.

Listing 3.12: Minimale XML-Instanz mit zyklischem Pfad

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE a
3 [ <!ELEMENT a EMPTY>
4   <!ATTLIST a id ID #IMPLIED idref IDREF #IMPLIED> ]>
5 <a id="a" idref="a"/>
```

Abbildung 3.10: Graphen-Darstellung der minimalen XML-Instanz mit zyklischem Pfad

Die Beurteilung, ob XML-Instanzen das formale Modell eines (geordneten, gerichteten) Baumes oder das eines Graphen nutzen, ist nicht ganz einfach. Ein wesentliches Unterscheidungskriterium ist das Vorliegen einer Grammatik, da diese benötigt wird, um Attribute als vom ID-Token-Typ bzw. IDREF-/IDREFS-Token-Typ auszuzeichnen (vgl. die Ausführungen im Abschnitt 3.4).[29]

Bei reiner Betrachtung einer Instanz (d. h. ohne Berücksichtigung einer Dokumentgrammatik, die für eine Typisierung von ID/IDREF/IDREFS-Attributen oder anderen lokalen Integritätsmerkmalen zwingend benötigt wird), erfüllt jede XML-Instanz die Anforderungen an einen geordneten (im Sinne der Abfolge der Knoten) und gerichteten (im Sinne der Dominanzbeziehung zwischen Eltern- und Kindknoten) Baum. Wird argumentiert, dass die durch ID/IDREF/IDREFS zusätzlich eingeführte Link-Kanten (es wird kein weiteres Nichtterminal im Baum eingefügt) gleichwertig sind zu den gerichteten

[29] Mit der Spezifikation XML:ID (Marsh, Veillard *et al.* 2005) besteht zwar die Möglichkeit, das generische Attribute `xml:id` (und nur dieses) als vom ID-Token-Typ zu deklarieren, ohne dass eine Dokumentgrammatik vorliegen muss, allerdings fehlt ohne Vorliegen einer solchen die Möglichkeit, per IDREF/IDREFS darauf zu verweisen – womit keine zusätzliche Kante eingefügt werden kann.

Kanten (Baumkanten), die durch die Dominanz-Beziehung erzeugt werden, geht das formale Modell über die Möglichkeiten eines Baumes und sogar eines DAG hinaus, da auch zyklische Pfade im Baum möglich sind (sofern eine Link-Kante ausgehend von einem Kindknoten hin zum Elternknoten etabliert wird). Das resultierende formale Modell entspricht damit einem gerichteten Graphen (DG). Bereits Lobin (2000, S. 61ff. und 66ff.) zeigt, dass sich sowohl azyklische als auch Graphen, die nicht dem Zyklizitätsverbot unterliegen, als XML-Instanzen darstellen lassen.

Eine solche Klassifizierung des formalen Modells von XML-Instanzen wird auch von weiteren Aufsätzen (vor allem aus dem Bereich der Datenbanken) geteilt (u. a. in Polyzotis und Garofalakis 2002). Gou und Chirkova (2007) unterscheiden zwischen dem Basis-Datenmodell des Baums (der allerdings rekursive Pfade erlaubt), das vorliegt, sofern auf Link-Kanten verzichtet wird, und dem erweiterten Datenmodell in Form von DAGs und allgemeinen Graphen, *General Graphs* – wobei letztere abhängig vom Vorhandensein zyklischer Pfade sind. Møller und Schwartzbach (2007); Møller und Schwartzbach (2011) bezeichnen das Datenmodell schlicht und einfach als *XML Graph* (vormals *Summary Graph*).

Argumentieren wir, dass eine über Link-Kanten eingeführte Beziehung zwischen zwei (oder mehreren) Knoten von einer anderen Qualität ist, da sie (1) abhängig ist von einer Dokumentgrammatik und (2) eine konstante Grenze hat (während die Menge von Bäumen zu einer Dokumentgrammatik prinzipiell unendlich sein kann, ist die Menge von Link-Kanten immer finit und begrenzt auf einen vollständig und konkret vorliegenden Baum), so lässt sich weiterhin das formale Modell einer XML-Instanz als Baum bezeichnen. In der Praxis werden Baumkanten in XML-Instanzen zuerst verarbeitet, um eine lineare Ordnung der Knoten und Blätter zu gewährleisten. Die zusätzlichen Link-Kanten werden in einem zweiten Schritt eingezogen, was für eine Unterscheidung spricht.

Dehmer *et al.* (2007) definieren diese Form von gerichteten Graphen (allerdings in einem anderen Zusammenhang) als generalisierte Bäume (*Generalized Tree*, GT). Ein GT ist ein gewurzelter gerichteter Graph, der über zusätzliche äußere Kanten (Link-Kanten) verfügen kann. Mehler (2009) klassifiziert darüber hinaus noch die Kanten, wobei in validen XML-Instanzen (also unter Einbeziehung eines Schemas) auf die hier definierten externen Kanten verzichtet wird.[30] Der Begriff ist nach Ansicht des Verfassers dieser Arbeit etwas unglücklich gewählt, da er eine verallgemeinerte Form des Baums suggeriert, während der Baum erweitert wird (um die Link-Kanten). Nahe liegender ist daher der Term des erweiterten Baums (*eXtended Tree*, XT). Ein solcher ist geordnet (durch die Präzedenzrelation der Knoten), verfügt über gerichtete Kanten (sowohl in Form der Dominanzbeziehung zwischen Eltern- und Kindknoten als auch der Link-Kanten) und eine einzige Wurzel. Die Erweiterung besteht durch die Einbeziehung zusätzlicher gerichteter Link-Kanten, die zyklisch sein können und auch auf den Wurzelknoten verweise können.

Während in der Informatik die allgemeinen Graphen weit verbreitet sind, herrschen in der Linguistik noch Baum-basierte Modelle vor (z. B. Phrasenstrukturbäume oder Rhetorische-Strukturbäume). Dennoch gibt es auch hier Ansätze, Graphen-basierte Mo-

[30] Sofern nicht noch weitere XML-begleitende Spezifikationen wie z. B. XLink zum Einsatz kommen.

delle stärker zu etablieren, wie Mehler (2009) anmerkt. Als Beispiele für den Einsatz von Graphen-Modellen in der Texttechnologie kann unter anderem Freisler (1994) angeführt werden, der sie als formales Modell von Hypertexten einführt, oder auch Widdows und Dorow (2002), die Graphen nutzen, um semantisches Wissen zu extrahieren und zu strukturieren, wobei Wörter als Knoten und Relationen als Kanten repräsentiert werden. Nastase und Szpakowicz (2006) präsentieren einen Algorithmus, der die Abbildung von Graphen zueinander verwendet, um semantische Relationen zuweisen zu können.

3.3.4 Weitere formale Modelle

Nicht auf XML-beruhende Metasprachen sind nicht an das formale Modell des Baums gebunden und sind daher prädestiniert zur Darstellung überlappenden Datenstrukturen und multipler Annotationen. Das in SGML noch vorhandene optionale CONCUR-Merkmal, das theoretisch zwei oder mehr Annotationsebenen (jeweils durch eine eigene DTD definiert) über die gleichen Textdaten erlaubt, wird in XML nicht mehr unterstützt (vgl. Goldfarb 1991b, S. 177; Clark 1997, Abschnitt 1) – selbst vermeintlich vollständige SGML-Parser haben dieses Merkmal oft nicht implementiert.[31] Ein Grund dafür dürfte die nur minimal vorhandene Beschreibung innerhalb des Standards sein, wie Barnard, Hayter *et al.* (1988, S. 271) bemängeln:

> [...] the description of the CONCUR feature in the Standard is fundamentally flawed, to the extent that this feature cannot be implemented as it is currently described. [...] Before CONCUR can be implemented, the Standard must be modified. [...] We conclude that CONCUR must be eschewed until the Standard is changed..

Listing 3.13: Beispiel für das in SGML vorhandene CONCUR-Merkmal

```
1 <(LOG)text>
2  <title>T</title>
3  <(ANN)comment>A comment</(ANN)comment>
4 </(LOG)text>
```

Im Listing 3.13 ist das Element `text` Teil der DTD, die durch die Zeichenkette „LOG“ identifiziert wird, `comment` ist in der DTD mit dem Bezeichner „ANN“ deklariert, und das Element `title` ist in allen verwendeten Dokumentgrammatiken vorhanden. Weitere, praxisnähere Beispiele finden sich in einer früheren Fassung der TEI GUIDELINES (P3, S. 22 und S. 623).

Ein Grund dafür, dass sich CONCUR nicht durchsetzen konnte, liegt – neben der unzureichenden Dokumentation und aufwendigen Implementierung auf Parserseite – daran, dass CONCUR einige Einschränkungen mitbringt: Abgesehen davon, dass für jede Annotationsebene eine eigene DTD vorhanden sein muss, müssen Elemente, die nicht in allen Annotationsebenen enthalten sind, zwingend mit einem Präfix (ein in runden Klammern eingeschlossener Schemabezeichner, der direkt vor dem Elementnamen steht) versehen werden. Das kann vor allem dazu führen, dass durch das Hinzufügen einer neuen Annotation die bisher nicht mit einem Präfix versehenen (da in allen

[31] So schreibt DeRose (1997, S. 110):„Very few parsers support CONCUR (I only have heard of one).“

bis dahin genutzten DTDs vorhandenen) Elemente durch ein solches ergänzt werden müssen, sofern auch nur ein Element nicht in der neuen Annotationsebene enthalten ist (vgl. Barnard, Burnard *et al.* 1995, S. 217). Darüber hinaus spricht sich der Herausgeber der SGML-Spezifikation deutlich gegen die Nutzung des CONCUR-Merkmals für die Darstellung verschiedener Annotationsebenen (als Sichten auf einen Text) aus:

> I therefore recommend that CONCUR not be used to create multiple logical views of a document, such as verse-oriented and speech-oriented views of poetry. Hypertext links are more appropriate for such applications, and can be created easily by taking advantages of unique identifiers and external references. (Goldfarb 1991b, S. 304f.).

Zu guter Letzt können mittels CONCUR auch keine Fälle von *Self Overlap* (vgl. Abschnitt 3.3.2 auf Seite 37) erfasst werden, da überlappende Datenstrukturen nur für Elemente aus unterschiedlichen DTDs erlaubt sind (vgl. DeRose 1997, S. 110f.). Interessanterweise findet sich das CONCUR-Merkmal in den TEI GUIDELINES (Abschnitt 31.1, „Concurrent Markup of Multiple Hierarchies“ P3, S. 623) bis zur P3 wieder, allerdings wird auch hier bereits im einleitenden Abschnitt darauf hingewiesen, dass eine Verwendung gut bedacht sein sollte:

> CONCUR is an optional feature of SGML, and not all available SGML software systems support it, while those which do, do not always do so according to the letter of the standard. For that reason, if for no other, wherever these Guidelines have identified a potential application of CONCUR, they also invariably suggest alternative methods as well. (P3, S. 24).

Eine Re-Implementierung von CONCUR in XML (XCONCUR, ehemals MULAX, vgl. Hilbert *et al.* 2005; Schonefeld und Witt 2006; Schonefeld 2007; Schonefeld 2008) zur Speicherung multipler Annotationen ist im Rahmen einer wissenschaftlichen Abschlussarbeit entwickelt worden und nutzt das formale Modell des *Multi-Rooted Trees*. Durch die Aufhebung der Restriktion einer einzigen Dokumentwurzel können überlappende Annotationsebenen in einem einzigen Dokument gespeichert werden (vgl. Witt, Schonefeld *et al.* 2007). Mit XCONCUR-CL ist ein Constraint-basierter Validierungsmechanismus skizziert worden (vgl. Schonefeld 2007, und Abschnitt 3.4), und das in Schonefeld (2008) vorgestellte XCONCUR SAX-API stellt zumindest rudimentäre Schnittstellen zur Verarbeitung multipler Annotationen zur Verfügung. Die weitere Entwicklung ruht z. Zt. allerdings.

Sowohl mit Hilfe von CONCUR als auch mit XCONCUR sind massiv multiple Annotation (z. B. mit fünf oder sechs verschiedenen Annotationsebenen) zumindest manuell nicht mehr sinnvoll nutzbar, womit der Vorteil der verschachtelten Inline-Annotation – die gleichzeitige Sichtbarkeit multipler Annotationsgrenzen – eingebüsst wird.

Weder Baum noch Graphen nutzt die *Range Algebra* (vgl. Nicol 2002b; Nicol 2002a). Als alternatives Modell für Texte und deren Annotationen können hiermit sowohl implizite als auch explizite Strukturen abgebildet werden (als Beispiel für eine implizite Struktur können Kreuzverweise, für eine explizite die bereits erwähnte Containment-Relation

gelten). Während Baum-basierte Strukturen und deren Algorithmen zur Validierung den vollständigen Baum als Ausgangspunkt nehmen, können auf *Range Algebra* basierte Verfahren auch Teil- und unvollständige Strukturen validieren. Allerdings ist die in Nicol (2002b) vorgestellte *Core Range Algebra* nicht ausdrucksstark genug, um sämtliche strukturellen Möglichkeiten von XML abzubilden; so ist das Vergeben eines Labels an einem markierten Bereich – analog z. B. zu einem Elementnamen in XML – nicht möglich, wohingegen überlappende Strukturen realisierbar sind. Die *Attributed Range Algebra* (Nicol 2002a) stellt eine Erweiterung dar, die die Attribuierung von Bereichen als basaler Form der Auszeichnung von Text erlaubt. Eine daran angelehnte Implementierung ist die Metasprache LAYERED MARKUP AND ANNOTATION LANGUAGE (LMNL, vgl. Tennison 2002; Piez 2004; Cowan, Tennison *et al.* 2006) – wobei ein vollständig generalisiertes abstraktes Datenmodell noch nicht vorliegt (vgl. auch Piez 2008). LMNL geht von einer Textebene als Basis aus, die aus eine Abfolge von null oder mehr Unicode-Zeichen bzw. Atomen besteht – wobei ein Atom durch eine andere Notation repräsentiert wird (auch Grafiken oder andere Multimedia-Objekte sind denkbar). Über diese Basisebene kann eine beliebige Anzahl an so genannten Bereichen (*Ranges*) aufgespannt werden. Ein Bereich enthält die Zeichen, die zwischen seinem Start-Tag und seinem End-Tag liegen. Jeder Bereich hat einen Namen, einen (evtl. impliziten) Bezeichner (der prinzipiell nicht Teil des Datenmodells ist, vgl. Cowan, Tennison *et al.* 2006) und Start- und Endkoordinaten, welche durch einen Offset und eine Länge über die Textebene dieses Bereiches dargestellt werden. Fehlende Bezeichner erlauben anonyme Bereiche – ähnlich wie bei den von Di Iorio *et al.* (2009) vorgestellten und in Abschnitt 3.3.3.1 diskutierten e-GODDAGs. Bereiche können prinzipiell mit anderen Bereichen überlappen (was auch gemeinsame Start-/Endpunkte in der Zeichenkette mit einschließt). Innerhalb eines Bereichs können beliebig viele Annotationen vorkommen. Eine Annotation hat eine Position (in Relation zu anderen Annotationen ihres Bereichs, d. h., Annotationen können im Gegensatz zu Bereichen nicht überlappen), einen Namen, der innerhalb des Bereichs nicht eindeutig sein muss, und eine Textebene, die wiederum selbst andere Bereiche beinhalten kann. Darüber hinaus können Annotationen andere Annotationen beinhalten (Meta-Annotationen) und Annotationen auch nur in End-Koordinaten (sprich: End-Tags) auftreten. Ein Beispiel für eine LMNL-Annotation mit Überlappungen (aus Piez 2008) ist in Listing 3.14 sichtbar:

Listing 3.14: LMNL-Beispiel mit Überlappungen

```
1 [excerpt [source}The Housekeeper{] [author}Robert Frost{]}
2 [s}[l [n}144{n]}He manages to keep the upper hand{l]
3 [l [n}145{n]}On his own farm.{s] [s}He's boss.{s] [s}But as to hens:{l]
4 [l [n}146{n]}We fence our flowers in and the hens range.{l]{s]
5 {excerpt]
```

Gegenüber anderen diskutierten Modellen ist die Tatsache bemerkenswert, dass LMNL keine Hierarchien zwischen Bereichen kennt, und damit die Relationen *Containment* und *Dominance* nicht abgebildet werden können: Die in den Listings 3.15 auf der nächsten Seite und 3.16 auf der nächsten Seite gezeigten beiden LMNL-Istanzen haben das gleiche Datenmodell. Aktuell gibt es zwar Bestrebungen seitens der Entwickler,

diesen Umstand zu ändern, bislang allerdings ohne Ergebnis.[32]

Listing 3.15: LMNL-Beispiel A (Sawtooth-Syntax)

```
1 [a}[b}this is a and b{b]{a]
```

Listing 3.16: LMNL-Beispiel B (Sawtooth-Syntax)

```
1 [b}[a}this is a and b{a]{b]
```

Im Gegensatz zu GODDAGs, die zumindest ursprünglich oftmals im Zusammenhang mit der Serialisierung in Form von MECS bzw. TexMECS genannt werden, spezifiziert LMNL bewusst nur ein Datenmodell, die Syntax ist nicht fest vorgegeben. So gibt es mit Canonical LMNL in XML (CLIX, ehemals bekannt als HORSE, Hierarchy-Obfuscating Really Spiffy Encoding DeRose 2004; Bauman 2005), ECLIX (extended CLIX), sowie xLMNL eine XML-Repräsentation für LMNL neben der LMNL-eigenen Sawtooth-Syntax. CLIX und ECLIX nutzen *XML Milestones* (vgl. Abschnitt 3.3.2) als Start- und Endpunkte für Bereiche, und *Trojan Milestones*, d. h., die Umwidmung beliebiger Elemente als Milestone-Marker, indem Start- und Endattribute hinzugefügt werden, so dass der semantische Gehalt erhalten bleibt, während xLMNL eine Standoff-Notation verwendet.

Weitere Arbeiten, die sich mit alternativen formalen Modellen befassen (allerdings im sehr engen Kontext von XML), und der Vollständigkeit halber nicht unerwähnt bleiben sollen, sind die bereits erwähnten Multi-Colored Trees (Jagadish *et al.* 2004), die vorrangig zum Zwecke der Abfragemöglichkeit von multiplen Annotationen entwickelt wurden und dazu XPath und XQuery erweitern.[33] Eine ebensolche Ergänzung des XDM (XQuery 1.0 and XPath 2.0 Data Model, Fernández *et al.* 2007; Berglund, Fernández *et al.* 2010) um die bereits genannten Delay Nodes diskutiert Le Maitre (2006) als ein weiteres Datenmodell. Beide Ansätze stellen nicht unbedingt eigenständige (und vollständige) formale Modelle dar, sondern modifizieren bestehende Ansätze. Der Ansatz der *Just-In-Time-Trees* (JITTs Durusau und O'Donnell 2002a) verneint nicht die Baumstruktur als solche, sieht aber das Problem darin, Auszeichnungen als statischen Baum von Metadaten über Zeichenketten anzusehen. Durch die Verlagerung der Erstellung einer Baumrepräsentation in die Verarbeitung, d. h., durch den während der Laufzeit dynamisch erzeugten Aufbau eines Baumes, können für jede Auszeichnung separate Bäume erstellt werden ohne Überlappungen zu erzeugen:

> The fundamental problem is that all prior methods treat markup as static trees of metadata about PCDATA. If that changes to: Trees are assertions about metadata, the fundamental difficulty of representing multiple trees in a single document instance resolves into a processing issue [...] The solution to representing multiple structures in a given text is to separate the assertion of trees from the markup in a text. (Durusau und O'Donnell 2002a)

[32] Vgl. den Beitrag von John Cowen vom 15.02.2011 auf der Mailingliste XML-dev, einzusehen unter `http://lists.xml.org/archives/xml-dev/201102/msg00084.html`, zuletzt abgerufen am 19.04.2012.

[33] Eine Diskussion der formalen Semantik von XQuery findet sich beispielsweise in Fankhauser (2001), eine entsprechend an diese angelehnte Implementierung wird durch Fankhauser und Lehti (2003) beschrieben.

Der in Durusau und O'Donnell (2002c) vorgestellte JITTs-Parser erlaubt die Verarbeitung multipler Annotationen in dynamischer Weise (und verleiht dem Ansatz seinem Namen).

In Durusau und O'Donnell (2005) lösen sich die Autoren noch weiter vom Baummodell: „There is no doubt that tree based processing of markup has been extremely useful in a variety of contexts. The question posed herein is whether that same utility can be matched by other data structures for processing markup." Der hier vorgestellte Ansatz nutzt eine relationale Datenbank in Verbindung mit einer Listenstruktur, um multiple Annotationen abbilden zu können: Elemente, Attribute und Primärdaten (Blätter im Baum) werden als Felder in einer Tabelle gespeichert, durch die Nutzung von Listen (anstelle von Bäumen) können auch überlappende Annotationen dargestellt werden (da Listen überlappen können). Ergebnisse von Abfragen auf die Datenbank können in grafischer Form repräsentiert werden, wobei sowohl der Ansatz der Datenspeicherung als auch die grafische Darstellung der Annotationen an die bereits angesprochene und in Witt (2004); Witt, Goecke, Sasaki *et al.* (2005) diskutierte Verwendung einer Prolog-Faktenbasis erinnert.

In Chatti, Calabretto *et al.* (2004) diskutieren die Autoren das *Multi-Structure Document Model* (MSDM), basierend auf Arbeiten von Abascal *et al.* (2003a); Abascal *et al.* (2003b). MSDM erlaubt, dass Teilbäume von Annotationen (Fragmente von Strukturen) gemeinsame Eltern haben. Ein *multi-structured Document* wird definiert durch eine Dokumentenstruktur (*Document Structure*, DS), eine Basisstruktur (*Base Structure*, BS) und der sogenannten Korrespondenz (*Correspondence*). Die DS stellt eine Sichtweise auf ein Dokument dar, vergleichbar mit einer Annotationsebene. Die nur intern sichtbare BS organisiert den Inhalt des Dokuments in disjunkte Einheiten. Durch die Korrespondenz, einer von der Dokumentenstruktur ausgehenden Relation zwischen zwei Strukturen ($DS \rightarrow BS$, $DS \rightarrow DS$), werden Elemente der DS mit dem Inhalt der Basisstruktur (im ersten Fall) oder implizite Relationen zwischen Einheiten der DS explizit ausgedrückt (im zweiten Fall). Mit MULTIX stellen Chatti, Kaouk *et al.* (2007) ein auf XML-basierendes Serialisierungsformat vor, in dem die entsprechenden Strukturen und Korrespondenzen aufgeführt werden. Die Basisstruktur wird in Form von `frag`-Elementen und zusammengefassten Fragmenten (in Form von `comp`-Elementen) realisiert. In den jeweiligen Dokumentstrukturen werden die entsprechenden Zeichenketten durch Referenzen auf Einheiten der Basisstruktur ersetzt. Die Korrespondenzen bestehen aus `clink`-Elementen mit entsprechenden `from`- und `to`-Attributen. Da weiterhin wohlgeformte XML-Instanzen genutzt werden, zeigen Chatti, Kaouk *et al.* (2007), dass sich mit Hilfe von XQUERY Abfragen über die multiplen Annotationen durchführen lassen. Portier und Calabretto (2009) greifen mit $MultiX^2$ MSDM als formales Modell wieder auf.

Bruno und Murisasco (2006) stellen mit MSXD ein formales Modell nebst XML-Serialisierung vor, das auf dem theoretischen Modell der *Hedge* (Hecke) basiert. Der Terminus „Hedge" wurde ursprünglich von Courcelle (1989) geprägt, und wurde vorrangig durch die darauf aufbauende Schemasprache RELAX NG (vgl. Abschnitt 3.4) bekannt. Ein Hedge ist eine geordnete Sequenz von (geordneten) Bäumen bzw. im Kontext der Verarbeitung von XML-Instanzen eine geordnete Abfolge von Elementen und konkreten Textdaten: „Informally, a hedge is a sequence of trees. In the XML terminology, a hedge is a sequence of elements possibly interevened [sic!] by character data (or types of character data); in particular, an XML document is a hedge." (Murata 1999, S. 1). Zuvor

hatte Murata den Begriff „Forest“ genutzt (vgl. Murata 1995), änderte ihn dann aber, um deutlich zu machen, dass eine Ordnung zwischen den Bäumen besteht – im Gegensatz zum bis dahin geläufigen „Forest“ als (ungeordnete) Menge von Bäumen.[34] Interessant an MSXD ist, dass das eigentliche multi-strukturierte Dokument nicht instanziiert wird, sondern nur über XQUERY-Erweiterungen verarbeitet wird (Bruno und Murisasco 2006, S. 147).

3.3.5 Beurteilung der formalen Modelle

Wie an dieser Stelle deutlich geworden sein sollte, ist eine eindeutige Festlegung auf ein formales Modell für XML-basierte Auszeichnungssprachen nicht ganz einfach. Die Befürworter des Baummodells führen allzu oft nur die Verschachtelung der Elemente an – die Möglichkeiten der von der Baumhierarchie unabhängigen Referenzierung von Knoten über Link-Kanten werden dabei vernachlässigt. Dass aber gerade diese Fähigkeiten dazu genutzt werden können, Graphen abzubilden (vgl. Stührenberg und Jettka 2009), bleibt häufig unberücksichtigt.

Zusammenfassend warnt auch Jelliffe (2010) deutlich davor, XML nur als Baumstruktur einzuschätzen: „Programmers and academics often think and theorize about XML as kind of tree data structure. And so indeed it is. But it is also allows much more: it is a series of different graph structures composed into or imposed on top of that tree.“ (Jelliffe 2010).

Für die Einrichtung von Link-Kanten über den XML-inhärenten Verknüpfungsmechanismus ist die Verwendung einer Dokumentgrammatik notwendig, da nur dann Informationen zu den Datentypen vorliegen. Anders ausgedrückt: Bei (lediglich) wohlgeformten XML-Instanzen hat das formale Modell des geordneten Baumes weiterhin Bestand. Unter Bezugnahme einer Dokumentgrammatik (sprich: bei validen XML-Instanzen) ist – je nach konkreter Grammatik – auch die Realisierung von Graphen möglich. Eine entsprechende Einschätzung findet sich bereits in Sperberg-McQueen 2004:

> For some purposes an SGML document is just an octet stream [...]. You can address it alternatively as a character stream; most of us prefer to do that. You can address it as a regular language in which everything is either a tag or content and you don’t care about the relations between tags. You can view it as a bracketed language [...]. You can view it as denoting the tree structure that is induced over the bracketed language. You can view it as defining a constrained tree structure – if you put a validator into your processor. You can pay attention to pointers – IDs and IDREFs – and say it defines a directed graph structure.

Tabelle 3.2 zeigt eine Übersicht über die in diesem Abschnitt diskutierten Metasprachen und formalen Modelle. Zur Erstellung einer Dokumentgrammatik stehen im Umfeld von XML verschiedene Grammatikformalismen zur Auswahl, die im nächsten Abschnitt diskutiert werden.

[34] Vgl. die Diskussion auf der Mailingliste relax-ng message vom 18.04.2001 einzusehen unter `http://lists.oasis-open.org/archives/relax-ng/200104/msg00033.html`.

Tabelle 3.2: Beurteilung der Metasprachen und formalen Modelle

	XML	XML+Schema	TEXMECS	LMNL	XCONCUR
Formales Modell	Baum	Baum/Hedge/ GODDAG/XT[1]	Clean r-GODDAG[2]	Attributed Range Algebra[3]	Mehrwurzeliger Baum
Status	W3C Recommendation			Forschungsprojekt	
Serialisierung	XML		Eigene	Eigene/XML[4]	Eigene
Hierarchien	Einfach[1]		Mehrfach	×	Mehrfach
Diskontinuierliche Einheiten	×	(✓)[1]	✓	✓	×
Constraint Language	×	Mehrere	Rabbit/Duck Grammar	CREOLE	XCONCUR-CL
Verbreitung		++	o[5]	−[6]	−−[7]

✓/× – Merkmal vorhanden/nicht vorhanden.
++ – sehr stark/+ – stark/o – zufriedenstellend/− – schwach/−− – sehr schwach.

[1] Je nach verwendeter Constraint Language (vgl. auch die Ausführungen in Teil III).
[2] Clean r-GODDAGs können 1:1 in TEXMECS serialisiert werden; zu weiteren GODDAG-Varianten vgl. die Ausführungen in Marcoux (2008b).
[3] Ein vollständiges formales Modell liegt nicht vor.
[4] Sawtooth-Syntax bzw. CLIX/ECLIX, xLMNL.
[5] Vorrangig durch den Einsatz im Projekt „Markup Languages for Complex Documents“ (MLCD).
[6] Kleine aber hochrangige Entwicklergemeinschaft.
[7] Weiterentwicklung ungewiss.

3.4 Grammatikformalismen für XML-Auszeichnungssprachen

Anders als SGML-Instanzen können XML-Instanzen hinsichtlich zweier Eigenschaften charakterisiert werden: der Wohlgeformtheit und der Validität (für eine Ausführung hinsichtlich der Unterschiede zwischen SGML und XML in dieser Hinsicht vgl. Lobin 2000, S. 69ff.). Wohlgeformtheit ist eine Minimalanforderung an XML-Instanzen, d. h., jede XML-Instanz muss wohlgeformt sein, um sich überhaupt als solche bezeichnen zu dürfen. Das ist dann gegeben, wenn alle folgenden Bedingungen erfüllt sind (*Well-Formedness Constraint*, vgl. Bray, Paoli und Sperberg-McQueen 1998, Abschnitt 2.1, „Well-Formed XML Documents“; Graham und Quin 1999, S. 119f. und S. 125ff.):

- Die Instanz enthält einen Prolog (bestehend aus einer optionalen XML-Deklaration, gefolgt von null oder mehr Kommentaren, *Processing Instructions* oder Leerzeichen) und genau ein XML-Element, gefolgt von null oder mehr Kommentaren, Processing Instructions oder Weißraum-Zeichen.[35] Da der Prolog selbst komplett leer sein kann, besteht die kleinste wohlgeformte XML-Instanz aus einem leeren Element.
- Das oben aufgeführte Element kann weiteren Inhalt (in Form von Kindelementen oder Text) beinhalten und darf selbst nicht in anderen Elementen enthalten sein, also nur als Wurzelelement genutzt werden.
- Für alle anderen Elemente (als Nachfahren des Wurzelelements) gilt: Sofern das Start-Tag innerhalb eines bestimmten Elements ist, muss auch das End-Tag in-

[35] Weißraum-Zeichen, d. h., Leerzeichen oder Steuerzeichen dürfen nicht vor dem Prolog stehen, sie sind aber am Ende der Instanz erlaubt.

nerhalb des selben Elements sein – hiermit wird gewährleistet, dass die Elemente korrekt in einander verschachtelt werden. Ebenso darf ein Element nicht Kind zweier Eltern-Elemente sein, oder anders ausgedrückt: Eltern-Kind-Relationen sind immer direkt auf einander folgend.

- Groß- und Kleinschreibung wird unterschieden.
- Attributwerte müssen mit Anführungszeichen (einfach oder doppelt) umschlossen werden.

Darüber hinaus gelten noch weitere Beschränkungen, u. a. in Bezug auf die Zeichen, die zur Definition von Element- und Attributnamen herangezogen werden können.

Für die zusätzliche Überprüfung auf Validität (Gültigkeit) wird eine Dokumentgrammatik benötigt. Eine Grammatik wird im Kontext von XML-Auszeichnungssprachen gemeinhin als Schema bezeichnet. Ein Schema ist in diesem Zusammenhang ganz allgemein ein Dokument, das eine XML-Auszeichnungssprache formal beschreibt. Einer der grundlegenden Aspekte einer solchen Beschreibung ist die Möglichkeit, Instanzen einer XML-Auszeichnungssprache mit ihrer Hilfe auf die Einhaltung folgender Bedingungen hin zu überprüfen (vgl. Walmsley 2002, S. 4ff.):

- Struktur der Auszeichnung: Die Verschachtelung der Elemente und die Struktur der Attribute stimmt mit der im Schema vorgegebenen überein.
- Reihenfolge: Die Reihenfolge der Elemente ist korrekt.
- Datentypen: Die Datentypen der Blätter im Baum (Textknoten und Attributwerte) entsprechen den im Schema gemachten Vorgaben.
- Integrität (spezifischer: hier von lokaler Integrität): Bei Nutzung von Datentypen, deren Wert dokumentweit einmalig sein muss bzw. die zur Herstellung von nicht auf die hierarchische Baumstruktur zurückzuführenden Beziehungen genutzt werden können (z. B. `ID/IDREF`), werden auch diese Eigenschaften überprüft.

Zusätzlich können Schemata für weitere Überprüfungen (so genannte *Business Rules*) herangezogen werden. Weitere Aufgaben sind die Dokumentation von Auszeichnungssprachen, die Bereitstellung von Vorgabewerten (*default value*) oder festen Werten in Instanzen sowie die Normalisierung von Weißraum-Zeichen je nach Datentyp:[36]

> Validation can be about checking the structure of XML documents. It can be about checking the content of each text node and attribute independently of each other (datatype checking). It can be about checking constraints on

[36] Streng genommen ist eine XML-Instanz im Sinne der XML-Spezifikation nur dann valide, wenn sie einer Dokumentgrammatik in Form einer DTD entspricht, da die XML-Spezifikation nur DTD als Validierungsmechanismen vorsieht. Andere Formalismen wurden separat entwickelt und selbst in neueren Fassungen der XML-Spezifikation (Bray, Paoli, Sperberg-McQueen, Maler und Yergeau 2008, Abschnitt 2.8, „Prolog and Document Type Declaration“) wurde davon abgesehen, eine Verknüpfung zu den sonstigen Validierungsmechanismen zu schaffen. Zu diesem Thema hat eine lebhafte Diskussion auf der Mailingliste XML-dev stattgefunden (nachzulesen unter `http://lists.xml.org/archives/xml-dev/200911/msg00019.html`, zuletzt abgerufen am 19.04.2012).

> relationships between nodes. It can be about checking constraints between nodes and external information such as lookup tables or links. It can be about checking business rules. Taken liberally, it can be almost anything else, even spell checking. (Van der Vlist 2003b, S. 4).

Einfach zusammengefasst, bringt Piez (2001) den Nutzen einer Validierung wie folgt auf den Punkt: „The intention or purpose of validation is to subject a document or data set to a test, to determine whether it conforms to a given set of external criteria“ (Piez 2001, S. 144).

Technisch gesehen kann man von XML-Schemata auch als einer Form der Transformation sprechen: Eine XML-Instanz wird in diesem Sinne als Eingabe einer Transformation (unter Zuhilfenahme eines *Schema Prozessors* oder validierenden Parsers) genutzt, die Ausgabe ist ein Gültigkeitsreport (*Validation Report*) nebst Rückgabewert und – optional – ein *Post Schema Validation Infoset* (PSVI), das die Eingabe-XML-Instanz (genauer: deren XML Infoset) mit weiteren Informationen anreichert (z. B. Vorgabewerte, Angaben zu den Datentypen, etc., vgl. van der Vlist 2003c, S. 368). Dieses PSVI kann allgemein auch als Normalisierung der XML-Instanz bezeichnet werden (vgl. Møller und Schwartzbach 2006b, S. 92).

3.4.1 Logische und technische Kriterien bei der Auswahl eines Grammatikformalismus

Bei Grammatiken, also auch bei XML-Schemata, ergibt sich prinzipiell die Problematik, dass bei einem Vergleich von angestrebter zu wirklicher Abdeckung (also Instanziierungen der Grammatik) die gewählten Constraints zu streng bzw. zu schwach sein können, was im ersten Fall die Abdeckung der Grammatik einschränkt (*Under-Generation*), im letzteren Fall zum Phänomen der Übergenerierung einer Grammatik (*Over-Generation*) führt. Anders ausgedrückt: Die von der Grammatik beschriebene Sprache ist kleiner oder größer als die abzubildende. Je schwächer die angelegten Constraints, desto mehr unzulässige Eingaben werden als gültig angesehen. Üblicherweise wird dieser Zusammenhang gerne bei der Beziehung zwischen einer konkreten Grammatik für eine XML-Auszeichnungssprache und der Menge der damit modellierbaren Instanzen gesehen:

> [S]chemas serving as a contract between data producers and data exchange partners have one role; schemas serving primarily to provide automatic annotation of the data have another; schemas which express our understanding of a corpus, in the form of a document grammar, have yet another. (Sperberg-McQueen 2007a, S. 9).

Diese Relation ist aber auch für das Paar Schema–Schemaformalismus interessant, da der zu Grunde liegende Formalismus (die Schemasprache) erst einige Konstruktionen zulässt, die über die Abdeckung der Grammatik entscheiden. Die im folgenden Abschnitt dieser Arbeit diskutierten Grammatikformalismen unterscheiden sich in einigen wesentlichen Punkten, so dass sich einzelne Inhaltsmodelle schwerer bzw. gar nicht

realisieren lassen können (vgl. auch Wilde 2006a, S. 92). Die Auswahl eines Grammatikformalismus, einer Schemasprache, kann ganz allgemein von verschiedenen Faktoren abhängen; oftmals wird praktischen Überlegungen wie einer guten Unterstützung durch Software der Vorrang gegenüber formalen Kriterien wie der Ausdrucksstärke (vgl. Abschnitt 3.4.2) gegeben. Auch Möglichkeiten der Modularisierung von Schemasprachen können durchaus einen Faktor bei der Entscheidung darstellen, da hierdurch auch die Wartbarkeit einer Dokumentgrammatik unmittelbar beeinflusst wird (Sperberg-McQueen 2007a, S. 11).

Die gebräuchlichsten XML-Schemasprachen sind die bereits in der XML-Spezifikation enthaltenen Dokumenttypdefinition (DTD), XML Schema, RELAX NG als Nachfolger der beiden Spezifikationen RELAX Core (ISO/IEC TR 22250-1:2002) und TREX (Clark 2001b) und Schematron (ISO/IEC 19757-3:2006). Sowohl RELAX NG als auch Schematron wurden im Rahmen des ISO-/IEC-Standards 19757, DSDL (Document Schema Definition Languages) standardisiert.

Ein wesentlicher Aspekt bei der Auswahl einer Schemasprache ist die leichte Erlernbarkeit: Während XML-DTD aufgrund seiner begrenzten Ausdrucksstärke und der einfachen Syntax sehr schell erlernbar ist, ist vor allem XML Schema recht umstritten. Das liegt in der Entwicklungshistorie begründet. Zum einen begannen die Arbeiten zur Standardisierung von XML Schema 1998 mit der Verfolgung mehrerer Ziele: Es sollten die Nachteile von XML-DTD beseitigt werden, d. h., die eigenständige Syntax, die unzureichende Typisierung von Daten und die fehlende Unterstützung von XML Namespaces (Bray, Hollander und Layman 1999; Bray *et al.* 2006a; Bray *et al.* 2006b). Zum anderen versuchte das Standardisierungskomitee des *W3C* (vgl. Abschnitt 2.1.2) die von DTD bekannten regulären Ausdrücke in Einklang zu bringen mit Objekt-Orientierung, die gerade aufgrund der erstarkenden Programmiersprache Java sehr gefragt war (Hosoya 2010, S. 2f.). Diese beiden Konzepte, das eine basierend auf Automatentheorie, das andere auf einem hierarchischen Modell, konnten nicht reibungslos zusammengeführt werden, was die Komplexität der Spezifikation von XML Schema in drei Teilen erklärt – und das, obwohl mit XML-Data (Layman *et al.* 1998), XML-Data Reduced (XDR, Frankston und Thompson 1998), Document Content Description for XML (DCD, Bray, Frankston *et al.* 1998), SOX (Schema for Object-Oriented XML, Fuchs, Maloney *et al.* 1998; Davidson *et al.* 1999) und der Document Definition Markup Language (DDML, Bourret *et al.* 1999) entsprechende Vorarbeiten geleistet waren.

RELAX NG dagegen wurde innerhalb recht kurzer Zeit aufgrund seines einfachen und klaren Aufbaus als Internationaler Standard verabschiedet und wird mehr und mehr eingesetzt (vgl. Abschnitt 3.4.5). Es bleibt abzuwarten, inwieweit die erst kurzzeitig in ihrer finalen Fassung verabschiedete neue Version von XML Schema, 1.1 (Peterson *et al.* 2012; Gao *et al.* 2012), diesen Trend beeinflussen kann.

Die in Abschnitt 3.3.4 diskutierten alternativen Metasprachen TexMECS, LMNL und XCONCUR besitzen eigene Constraint Languages, namentlich Rabbit/Duck Grammars (u. a. für TexMECS, vgl. Sperberg-McQueen 2006; Sperberg-McQueen und Huitfeldt 2008b), CREOLE (Composable Regular Expressions for Overlapping Languages etc., eine Erweiterung von RELAX NG, die u. a. zur Validierung von LMNL-Instanzen genutzt werden kann, vgl. Tennison 2007) und XCONCUR-CL (XCONCUR-Constraint Language, eine regelbasierte Sprache ähnlich Schematron, vgl. Schonefeld 2007), die

aber nicht weiter Gegenstand dieser Arbeit sein sollen und nur der Vollständigkeit halber hier aufgeführt sind.

3.4.2 Formale Kriterien zur Auswahl eines Grammatikformalismus

XML-basierte Auszeichnungssprachen können mit einer Reihe von Formalismen definiert werden, d. h., mit formalen Sprachen zur Definition von Schemata (Møller und Schwartzbach 2006b, S. 92). Diese Grammatikformalismen lassen sich nach verschiedenen Kriterien unterscheiden, z. B. anhand der verwendeten Syntax (einige verwenden ebenfalls die Syntax von XML-Instanzen, andere eine separate) oder aber ihres Beschreibungsinventars, dass zur Entwicklung von XML-Anwendungen eingesetzt werden kann. In Bezug auf Auszeichnungssprachen lassen sich diese Grammatikformalismen zusätzlich dazu nutzen, Aussagen über deren Mächtigkeit in Bezug auf ihre Sprachenklasse (Ausdrucksmächtigkeit) zu treffen, da es auch Unterschiede in der formalen Ausdrucksstärke zwischen den einzelnen Grammatikformalismen gibt.

Neben den bereits im Abschnitt 3.4.1 genannten weit verbreiteten Grammatikformalismen DTD, XSD und RELAX NG gab und gibt es auch noch weitere Ansätze, wie z. B. mehrere Vorschläge für eine erweiterte DTD-Spezifikation (vgl. Buck *et al.* 2000; Papakonstantinou und Vianu 2000; Vitali *et al.* 2003; Balmin *et al.* 2004; Fiorello *et al.* 2004), Document Structure Description (DSD bzw. DSD2, vgl. Klarlund *et al.* 1999; Klarlund *et al.* 2002; Møller 2002), Constraint Language in XML (CLiX, vgl. Nentwich 2005, keine erkennbare aktive Entwicklung mehr – nicht zu verwechseln mit CLIX aus Abschnitt 3.3.4), Content Assembly Mechanism (CAM, vgl. Carey 2009), sowie die Meta-Validierung unter Zuhilfenahme der Namespace-based Validation Dispatching Language als Teil des Multi-Part-Standard ISO/IEC 19757 (ISO/IEC 19757-4:2006). Die auf Prädikatenlogik erster Stufe basierende regelbasierte Sprache xlinkit (vgl. Nentwich, Emmerich *et al.* 2001; Nentwich, Capra *et al.* 2002) wird anscheinend nicht mehr weiter entwickelt.

Prinzipiell lassen sich XML-Schemasprachen in zwei Klassen einteilen: Während DTD, XSD und RELAX NG *grammatikbasierende* Schemasprachen darstellen, bilden Schematron[37], CLiX, xlinkit und DSD2 die Klasse der *Constraint-* oder *regelbasierten Sprachen*, die mit Hilfe von Pfadausdrücken (typischerweise mittels XPath) Aussagen darüber machen, ob eine XML-Instanz den festgelegten Regularien entspricht.[38] Eine sinnvolle Unterscheidung beider Klassen findet sich in sowohl in van der Vlist (2003c); van der Vlist (2004), als auch in Costello und Simmons (2008, Abschnitt „Two Types of XML Validation“):

> A grammar-based schema language specifies the structure and contents of elements and attributes in an XML instance document. For example, a grammar-based schema language can specify the presence and order of

[37] Eine kurze Diskussion hinsichtlich der Klassifizierung von Schematron als regelbasierter Sprache findet sich in Jelliffe (2009).

[38] Bauman (2008) diskutiert umfassend die verschiedenen Möglichkeiten, Constraints auszudrücken, und die Vor- und Nachteile, die der Einsatz geschlossener bzw. offener Constraint Languages mit sich bringen kann.

> elements in an XML instance document, the number of occurrences of each element, and the contents and datatype of each element and attribute. A rule-based schema language specifies the relationships that must hold between the elements and attributes in an XML instance document. For example, a rule-based schema language can specify that the value of certain elements must conform to a rule or algorithm.

Van der Vlist (2003b, S. 9) unterscheidet sogar zwischen drei Klassen: den regelbasierten (als Beispiel wird auch hier SCHEMATRON genannt), den beschreibenden („Constraints may be expressed as a thorough description of each element and attribute“, als Beispiele werden DTD und XSD genannt), und den Formalismen, die mit Hilfe von Mustern (*Pattern*) die Eigenschaften einer Sprache festlegen: „Constraints may be expressed as patterns. Patterns are used to match the structures of permissible elements, attributes, and text nodes, much as the regular expressions used in programming can be used to match characters in text“. Zur letztgenannten Kategorie zählt der Autor RELAX NG.

Festzuhalten bleibt, dass regelbasierte Sprachen wie SCHEMATRON sich von den anderen beiden Klassen, konkret: von DTD, XSD und RELAX NG, deutlich unterscheiden: Während die drei letztgenannten die Struktur gültiger Dokumente beschreiben (van der Vlist 2003c, spricht von „geschlossenen Schemasprachen“), ist erstere „die einzige XML-Schemasprache, die ihre Validierung nicht auf ein Modell der Klasse von Dokumenten, die als gültig betrachtet werden, aufbaut“ (van der Vlist 2003c, S. 369).

In Bezug auf die unterstützten Merkmale von XML-Schemasprachen gibt es eine Reihe von Arbeiten, die einen Überblick verschaffen: Schon vor Veröffentlichung der endgültigen Version von XML SCHEMA 1.0 vergleicht Walsh (1999) die neue Spezifikation mit den bisher allein verfügbaren DTD. D. Lee und Chu (2000) untersuchen insgesamt sechs zum damaligen Zeitpunkt aktuelle XML-Schemasprachen (DTD, XML SCHEMA, XDR, SOX, SCHEMATRON und DSD) und vergleichen sie auf ihre Konstrukte und Merkmale, darunter Syntax, Unterstützung von Namensräumen, Modularität, enthaltene Datentypen und die Möglichkeit der Definition eigener Datentypen, Kardinalität und Okkurrenzindikatoren sowie *Co-occurrence Constraints* (auch *Co-Constraints*, vgl. Abschnitt 3.4.4.4). Darüber hinaus ordnen die Autoren die Formalismen nach formaler Ausdrucksstärke und kommen zu dem Ergebnis, dass DTD als Klasse-1-Schemasprache (d. h. mit geringem Sprachumfang) und XML SCHEMA und SCHEMATRON jeweils in der höchsten Klasse 3 einzuordnen ist. Jelliffe (2001) vergleicht DTD, XML SCHEMA, RELAX NG, SCHEMATRON, DSDL (damals noch eine eigenständige Schemasprache), XLINKIT, EXAMPLOTRON (van der Vlist 2003a) und das minimalistische, auf partieller Ordnung basierende HOOK (Jelliffe 2002) anhand der Punkte *Encoding*, *Linking*, *Infoset*, *Structures*, *Static Datatyping*, *Local Reference Integrity*, *Web Reference Integrity*, Co-occurrence Constraints und *Value Defaulting* und macht die Stärken und Schwächen der jeweiligen Grammatikformalismen deutlich.

Van der Vlist (2001) untersucht ebenfalls eine Reihe von XML-Schemasprachen, darunter DTD, XML SCHEMA, RELAX NG, SCHEMATRON und EXAMPLOTRON. Unterschiede manifestieren sich u. a. an der Erzeugung eines PSVI – bei der Validierung mittels DTD und XML SCHEMA wird ein Post Schema Validation Infoset erstellt, während die

Validierung mit Hilfe von RELAX NG ohne auskommt. SCHEMATRON erzeugt nicht direkt ein PSVI und zum Zeitpunkt der Untersuchung sollte EXAMPLOTRON noch um ein solches erweitert werden. Integrität, d. h., die Gültigkeit lokaler Verknüpfungen zwischen Knoten innerhalb eines Baumes, lässt sich in DTD (durch Attribute vom Token-Typ `ID` bzw. `IDREF/IDREFS`) und XML SCHEMA (sowohl durch die Datentypen `xs:ID` bzw. `xs:IDREF/xs:IDREFS` als auch durch `xs:unique`, `xs:key` bzw. `xs:keyref`) sicher stellen, während die anderen betrachteten Grammatikformalismen diese nicht (oder nicht unmittelbar) unterstützen. Zu beachten ist, dass sich mittels `ID` und `IDREF` eine 1:1-Beziehung zwischen zwei Knoten im Baum, mittels `ID` und `IDREFS` 1:*n*-Beziehungen herstellen lassen – wobei über die Art der Beziehung nichts ausgesagt werden kann (außer evtl. implizit durch den Namen des Trägers des jeweiligen Typ). Ebenfalls interessant in dieser Aufstellung ist die Gegenüberstellung in Bezug auf Support durch Software und Flexibilität im Einsatz, die trotz des Alters des Artikels auch noch heute Gültigkeit besitzt (vgl. Abschnitt 3.4.5).

Møller und Schwartzbach (2006b) stellen ebenfalls verschiedene Grammatikformalismen gegenüber: DTD, XML SCHEMA, DSD 2.0 und RELAX NG. Hervorzuheben an dieser Arbeit ist – neben der Ausführlichkeit – die präzise Aufstellung der Nachteile der XML-Schemasprachen DTD und XML SCHEMA. Als ein Beispiel sei hier nur auf die in XML SCHEMA verwirrende (aber erlaubte) Nutzung so genannter *unqualified locals* verwiesen, was die Autoren dazu veranlasst, Best-Practice-Regeln für die Nutzung von XML SCHEMA als Grammatikformalismus zur Definition einer Auszeichnungssprache anzuführen (Møller und Schwartzbach 2006b, S. 158), deren Befolgung die ursprüngliche Ausdrucksstärke von XML SCHEMA reduziert.[39] Ähnliche Ratschläge finden sich bereits in Kawaguchi (2001b).

Ansari *et al.* (2009) untersuchen dieselben vier Grammatikformalismen, unterlassen aber eine Wertung, so dass die Darstellung insgesamt eher oberflächlich bleibt.

Dagegen stellt Manguinhas (2009) die drei wesentlichen grammatikbasierten XML-Schemasprachen DTD, XML SCHEMA und RELAX NG, sowie die regelbasierte Sprache SCHEMATRON gegenüber und geht dabei auch kurz auf die von Murata, D. Lee, Mani und Kawaguchi (2005) aufgestellte Taxonomie nach formalen Gesichtspunkten ein (vgl. auch die folgenden Abschnitte).[40] Abschließend bewertet Manguinhas die Grammatikformalismen nach den Aspekten Ausdrucksstärke, Wiederverwendbarkeit/Modularität, Kompaktheit, Komplexität (im Sinne von Aufwand der Einarbeitung) und Erweiterbarkeit, wobei dem Autor nicht in allen Punkten zu folgen ist. So ist z. B. diskussionswürdig, SCHEMATRON als die am schwersten zu lernende der vorgestellten Formalismen einzustufen und unter Berücksichtigung der alternativen Syntax von RELAX NG (Compact

[39] Elemente, die *lokal* in einem XSD deklariert wurden, dessen Attribut `elementFormDefault` des Wurzelelements den Vorgabewert *unqualified* hat, sind nicht Namensraum-qualifiziert (*Namespace qualified*). Da global deklarierte Elemente und Attribute immer Namensraum-qualifiziert sind, und nur bei lokal delarierten Elemente beide Möglichkeiten erlaubt sind, führt dieser Umstand oftmals zu Verwirrung sowohl bei XSD-Autoren als auch bei der Nutzung in der Instanz. Eine Untersuchung bezüglich der Verbreitung einiger XSD-Merkmale findet sich in Lämmel *et al.* (2005).

[40] Interessant ist in diesem Zusammenhang, dass in Manguinhas (2009, S. 663) RELAX NG als Restrained-Competition Tree Grammar klassifiziert wird, während in Murata, D. Lee, Mani und Kawaguchi (2005, S. 701) RELAX NG als Formalisierung einer Regular Tree Grammar charakterisiert wird. Stührenberg und Wurm (2010) stützen diese Ansicht.

Syntax, definiert in ISO/IEC 19757-2:2003/Amd 1:2006) könnte man diese Schemasprache als ebenso kompakt einstufen wie DTD.

Wie an den oben genannten Beispielen für XML-Schemasprachen schon deutlich geworden ist, ist eine gewisse Nähe einiger dieser Grammatikformalismen zu den formalen Sprachen bzw. formaler Logik durchaus gegeben (beispielsweise bei XLINKIT oder HOOK). So ist es nicht verwunderlich, dass neben den bereits diskutierten Ausführungen zu vorhandenen respektive fehlenden Merkmalen innerhalb der untersuchten Grammatikformalismen sich eine Reihe von Autoren mit den formalen Grundlagen (und Unterschieden) befasst hat. Zu nennen wären hier die Arbeiten von Brüggemann-Klein und Wood (1992); Brüggemann-Klein (1993a); Brüggemann-Klein und Wood (1997a); D. Lee, Mani *et al.* (2000); Hopcroft, Motwani *et al.* (2000); Rizzi (2001); Mani (2001); Murata, D. Lee und Mani (2001); Brüggemann-Klein und Wood (2002); Berstel und Boasson (2002); Sperberg-McQueen (2003); Klarlund, Schwentick *et al.* (2003); Brüggemann-Klein und Wood (2004); Murata, D. Lee, Mani und Kawaguchi (2005); Martens, Neven und Schwentick (2005); S. Lu *et al.* (2005); Kilpeläinen und Tuhkanen (2007); Comon *et al.* (2008); Martens, Neven und Schwentick (2009); Gelade, Martens *et al.* (2009); Gelade, Idziaszek *et al.* (2010), die sich alle mit der Beziehung zwischen Grammatiken bzw. Automaten und XML-Schemasprachen beschäftigen. Kracht (2011) erweitert diese Arbeiten, in dem er eine modale Logik verwendet, um eine Semantik für XML-Instanzen zur Verfügung zu stellen. Darüber hinaus bezieht er XPATH zur Adressierung von Informationseinheiten in XML-Instanzen in die Diskussion mit ein. Auch Benedikt und Koch (2009) untersuchen die Ausdrucksmächtigkeit von XPATH.

Beim Versuch, mathematisch-logische Modelle und Formalismen in Bezug zu XML-Auszeichnungssprachen (sowohl auf Seite der Instanz als auch auf Seite der verwendeten Schemasprache) zu setzen, ist zunächst die teilweise unterschiedliche Verwendung in der Terminologie zu beachten. In der Literatur zu XML finden sich Darstellungen ähnlich der Abbildung 3.3 auf Seite 35, in der die Knoten eines Baumes das jeweilige Knotenlabel direkt verwenden, was zu der irrigen Annahme führen kann, Knoten und Knotenlabel als Einheit anzusehen. Das ist darauf zurückzuführen, dass der ursprünglich verwendete Formalismus zur Beschreibung von XML-Dokumentgrammatiken, DTD, als erweiterte kontextfreie Grammatiken (*Extended Context Free Grammar, ECFG*) angesehen werden kann: „A DTD is essentially a context-free grammar, with its own notation for describing the variables and productions. [...] Rather, the language for describing DTDs is essentially a CFG notation [...]“ (Hopcroft, Motwani *et al.* 2000, S. 199).

Als ECFG werden kontextfreie Grammatiken angesehen, in denen auf der rechten Seite der Produktionsregeln reguläre Ausdrücke (im Gegensatz zu endlichen Sprachen) erlaubt sind. Das bedeutet für die Deklaration eines Elements, dass das erlaubte Inhaltsmodell (die rechte Seite der Regel) durch einen regulären Ausdruck beschrieben werden kann, der entweder andere Elementnamen (*Generic Identifier*) oder reservierte Schlüsselwörter (wie z. B. `#PCDATA` oder `EMPTY`) beinhaltet. Analog dazu Brüggemann-Klein und Wood (1997b, S. 2):

> Document types in SGML are defined, essentially, by bracketed, extended context-free grammars [...]. The right-hand sides of productions, called model groups, are essentially regular expressions with two major differences.

> First, model groups allow three new operators ?, &, and +. Second, the model groups must be unambigiuous in the sense of Clause 11.2.4.3 of the standard.

Rizzi (2001, S. 107) erweitert diese Definition in Bezug auf XML-Inhaltsmodelle:

> [...] [S]everal authors [...] model DTDs as extended context-free grammars expressed in a notation that is similar to extended Backus-Naur form. In addition, the SGML standard allows the semantics of content models (the right-hand side of productions) to be modified by exceptions. [...] Unlike SGML DTDs, XML DTDs do not allow exceptions but proposals to incorporate some exception mechanisms in XML also are still somewhat under debate as a means of reducing the difficulty of translating SGML DTDs into XML DTDs.

Ebenso bejahen Klarlund, Schwentick *et al.* (2003, S. 13) den Zusammenhang zwischen ECFG und DTD:

> Ignoring the attributes for a moment, there is a simple but elegant connection between DTDs and context-free grammars, namely, each DTD corresponds to an *extended context-free grammar*, where productions may have regular expressions on their right-hand side. Then, an XML document is valid with respect to the DTD precisely when its associated tree is a correct derivation tree for that grammar.

Zur näheren Klassifizierung der Grammatik führen Brüggemann-Klein und Wood (1997b, S. 75) weiter aus:

> The grammars are both a restriction and a generalization of the bracketed grammars of Ginsburg and Harrison [...]: a restriction since each nonterminal has only one production; a generalization since the right-hand side of a production is a regular expression as opposed to a simple string.[41]

Neuere Abhandlungen (vgl. Murata, D. Lee und Mani 2001; Murata, D. Lee, Mani und Kawaguchi 2005; Møller und Schwartzbach 2006a) nutzen zur Klassifizierung von XML-Grammatikformalismen Baumgrammatiken.[42] Der wesentliche Unterschied zwischen (E)CFGs und (lokalen) Baumgrammatiken besteht darin, dass erstere Zeichenketten und letztere Bäume generieren (Gladky und Melčuk 1969, S. 1; Murata, D. Lee, Mani und Kawaguchi 2005, S. 662 und S. 686f.), darüber hinaus können Baumgrammatiken mehr als ein Start-Symbol haben (Brüggemann-Klein und Wood 1998, S. 7). Da es sich darüber hinaus um gelabelte Bäume handelt, lässt sich der Kontext anhand der geordneten Knotenfolge (Präzedenz-Relation) betrachten. Auf diese Weise lässt sich das Inhaltsmodell mit einem regulären Ausdruck beschreiben und mit Hilfe eines endlichen Automaten (*Finite State Machine*, FSM) verarbeiten. Für den weiteren Verlauf dieser Arbeit wird dieser Argumentation gefolgt, da mit dem Zur-Verfügung-Stehen von entsprechenden

[41] Das Zitat bezieht sich auf die in Ginsburg und Harrison (1967) diskutierten Ansätze.

[42] Für eine Einführung in das Konzept der Baumgrammatiken sei auf Gladky und Melčuk (1969), für eine ausführliche Behandlung auf Gécseg und Steinby (1997) verwiesen.

Validierungsalgorithmen für Baumgrammatiken auf Hilfskonstruktionen verzichtet werden kann (Murata, D. Lee, Mani und Kawaguchi 2005, S. 687f.).

Mit DTD als weit verbreiteten und ältesten Formalismus zur Erstellung von XML-Auszeichnungssprachen erstellte Schemata werden in Murata, D. Lee, Mani und Kawaguchi (2005) als lokale Baumgrammatiken (*Local Tree Grammar*) klassifiziert. In lokalen Baumgrammatiken besteht für jedes Knotenlabel genau eine Regel, die es erzeugt – was zu der oben angesprochenen Annahme führen kann, dass Knoten und -label als Einheit angesehen werden. Die Abbildung 3.11 verdeutlicht den Unterschied im Vergleich zur Abbildung 3.3 auf Seite 35: Die Label im Datenmodell von XML bezeichnen Knoten, nicht Kanten (Abiteboul *et al.* 2000, S. 33).

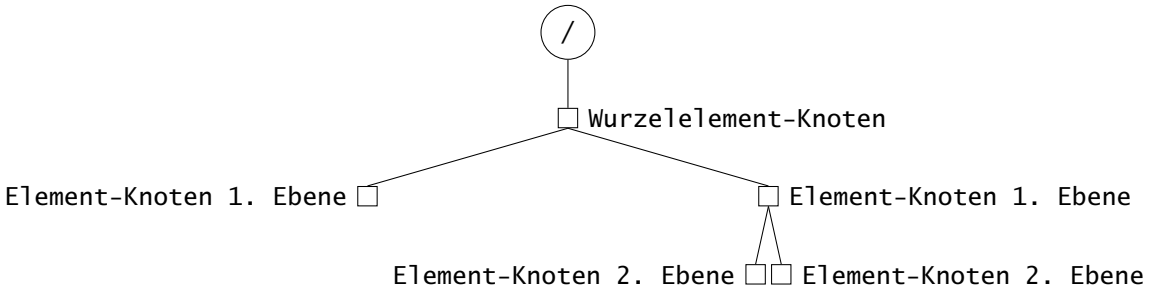


Abbildung 3.11: Darstellung zur Trennung von Knoten und -label

Abschnitt 3.4.4 befasst sich näher mit der Klassifizierung der unterschiedlichen Grammatikformalismen in Bezug auf ihre Ausdrucksstärke, der folgende Abschnitt 3.4.3 thematisiert das zentrale Konzept des Determinismus'.

3.4.3 Determinismus

Ein wesentlicher Aspekt im formalen Modell von XML Schemasprachen ist der des Determinismus' bezogen auf das Inhaltsmodell. Wie bereits in ISO 8879:1986, Abschnitt 11.2.4.3 festgelegt, werden Inhaltsmodelle von SGML- und XML-DTD künstlich beschränkt:

> The purpose of the content model is, in effect, to state an agreement between human beings and computer programs as to what will occur in a given element. Although a computer could make sense of a content model that SGML prohibits, a human is likely to encounter difficulties with it. (Goldfarb 1991b, S. 414f.).

Die SGML-Spezifikation definiert diese Einschränkung wie folgt: „A content model cannot be ambiguous; that is, an element or character string that occurs in the document instance must be able to satisfy only one primitive content token without looking ahead in the document instance." (ISO 8879:1986, Abschnitt 11.2.4.3).

Dieser *look-ahead* wurde verboten, um eine Situation gar nicht erst entstehen zu lassen, in der der Parser sehr weit vorausschauen muss, was zu einem erhöhten

Verarbeitungsaufwand (sowohl zeitlich als auch in Bezug auf den Speicherverbrauch) führen kann (van Herwijnen 1994, S. 184f.).

Zu beachten ist, dass die Definition von Nicht-Ambiguität (Unzweideutigkeit, *Unambiguity*) im Sinne von SGML und XML sich von der von regulären Ausdrücken unterscheidet. Aufbauend auf der Definition von Nicht-Ambiguität von Book *et al.* (1971) prägen Brüggemann-Klein und Wood (1997a, S. 183) dafür den Term *1-unambiguity/one-unambiguity* (etwa: 1-Schritt-eindeutig) und veranschaulichen diese Unterscheidung an folgendem Beispiel: Der reguläre Ausdruck $(a + b)^*a$ kann durch folgende, indizierte Schreibweise dargestellt werden: $(a_1 + b_1)^*a_2$. Durch Hinzufügen eines Positionsindex können mehrere Vorkommen ein und desselben Symbols an verschiedenen Stellen markiert werden – was im nächsten Schritt die Auflösung ambiger Ausdrücke erlaubt. Das Wort baa kann nun auf zweierlei Art erzeugt werden: Nach dem ersten b (erzeugt durch b_1) kann nur entschieden werden, ob das folgende a durch a_1 oder a_2 generiert wird – außer, man erlaubt den look-ahead um eine Position. Daher ist der Ausdruck $(a + b)^*a$ nicht-ambig im Sinne von Book *et al.* (1971), nicht aber 1-nicht-ambig im Sinne von Brüggemann-Klein und Wood (1997a) (d. h., er ist 1-ambig). Es kann aber der 1-nicht-ambige reguläre Ausdruck $b^*a(b^*a)^*$ genutzt werden, um Ausdrücke der gleichen Sprache zu erzeugen.

Hintergrund des Verbots ambiger Inhaltsmodelle in SGML- und XML-DTD (und in XML SCHEMA, vgl. die Ausführungen in Abschnitt 3.4.4) ist die Anwendbarkeit endlicher Automaten für die Überprüfung von Inhaltsmodellen (genauer: zur Kontextüberprüfung (*Context Check*) von *Model Groups*)[43]. Anhang H der SGML-Spezifikation („Theoretical Basis for the SGML Content Model“) führt dazu aus:

> Checking for conformance to a content model is essentially equivalent to the problem of recognizing (accepting or rejecting) regular expressions, and regular expressions therefore provide a useful theoretical basis for some aspects of content checking in SGML. It can be shown (by Kleene's theorem) that a regular expression is equivalent to a deterministic finite automaton (DFA). A parser could in theory, therefore, handle any *model group* by reducing it to a regular expression and constructing a DFA with state transition paths corresponding to the tokens of the *model group*. Practice, however, presents some difficulties. [...] A[nother] problem lies with the construction of the DFA. One method is first to construct a nondeterministic finite automaton (NFA) directly form the regular expression, then to derive a deterministic equivalent. Although this and other algorithms for DFA construction are non-polynomial and hardly intractable for the human-readable models envisaged by SGML, they may be too resource-intensive for some implementations. This International Standard avoids these problems by eliminating the need to construct a DFA. This it does by prohibiting models that are ambiguous or require „look-ahead“; [...] As a result, context checking

[43] Nach DeRose (1997, S. 231) sind die Begriffe *Content Model* und *Model Group* fast synonym zu verwenden – mit der Ausnahme, dass das Schlüsselwort `ANY` zwar als Inhaltsmodell angesehen wird, während eine Model Group aber das Vorhandensein eines geklammerten Ausdrucks inkl. Generic Identifier voraussetzt. Im Folgenden werden daher die Begriffe synonym verwendet.

> can be done by simplified algorithms that use only NFAs. (Goldfarb 1991b, S. 558f.).

Interessant am Verbot von „ambigiuous content models“ (DeRose 1997, S. 225) in der SGML-Spezifikation ist die Tatsache, dass das Vorliegen eines ambigen Inhaltsmodells in der DTD nicht generell zum Abbruch der Verarbeitung führt, erst beim Auftreten in der Instanz ist der Parser angehalten, den Fehler zu melden, so dass dieses längere Zeit unbemerkt bleiben kann (van Herwijnen 1994, S. 184). Eine ausführliche Betrachtung ambiger Inhaltsmodelle und Ambiguitäten durch *Tag Minimization* (vgl. Listing 3.2 auf Seite 31) findet sich in Graf (1988) – wobei nicht alle der aufgeführten Beispiele korrekt sind (vgl. McFadden und Wilcott 1989). Die Bezeichnung änderte sich in der XML-Spezifikation (im nicht-normativen Teil) zu „non-deterministic content models“ (Bray, Paoli und Sperberg-McQueen 1998, Anhang E, „Deterministic Content Models“), während XML SCHEMA zur Behandlung solcher Inhaltsmodelle explizit die *Unique Particle Attribution* (UPA) (im normativen Teil) einführt:

> A content model must be formed such that during validation of an element information item sequence, the particle contained directly, indirectly or implicitly therein with which to attempt to validate each item in the sequence in turn can be uniquely determined without examining the content or attributes of that item, and without any information about the items in the remainder of the sequence. (Thompson *et al.* 2001, Abschnitt 3.8.6, „Constraints on Model Group Schema Components“, Unterabschnitt „Schema Component Constraint: Unique Particle Attribution“).

Im Listing 3.17 kann die verarbeitende Software nach Parsen des Start-Tags des Elements a entweder das Element b (gefolgt von c), das Element c (gefolgt von d) oder aber das End-Tag von a erwarten, es gibt also jeweils nur eine Möglichkeit.

Listing 3.17: Deterministisches Inhaltsmodell

```
1 <!ELEMENT a ((b,c) | (c,d))?>
```

Anders im Listing 3.18: Hier ist unklar, ob nach dem Element b das Element c oder d folgt, die Auswahl des nächsten Partikels ist damit nicht eindeutig.

Listing 3.18: Nicht-deterministisches Inhaltsmodell

```
1 <!ELEMENT a ((b, c) | (b, d))?>
```

Abbildung 3.12 zeigt das entsprechende Zustandsdiagramm (vgl. auch Maler und El Andaloussi 1995, S. 223).

Sowohl XML-DTD als auch XML Schema lassen nur deterministische Inhaltsmodelle zu, eine Einschränkung, die oftmals nicht als solche wahrgenommen wird, da sich ein großer Teil der nicht-deterministischen Inhaltsmodelle auch als deterministische Inhaltsmodelle umschreiben lässt. Listing 3.19 zeigt die entsprechende Umschreibung des Inhaltsmodells aus Listing 3.18.

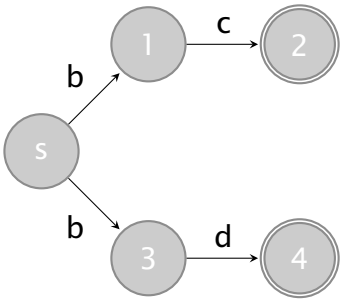


Abbildung 3.12: Zustandsdiagramm eines nicht-deterministischen endlichen Automaten (NFA) für das Inhaltsmodell aus Listing 3.18

Listing 3.19: Deterministische Umschreibung des Inhaltsmodells

```
1 <!ELEMENT a (b,(c|d))?>
```

Dieses umgeschriebene Inhaltsmodell lässt in der Instanz die gleichen Auswahlmöglichkeiten zu wie das Inhaltsmodell in Listing 3.18, da allerdings nach dem Start-Tag des Elements a entweder das Element b oder das End-Tag von a folgt, ist es deterministisch. Im Falle von b folgt anschließend entweder c oder d – auch diese Auswahl ist eindeutig und damit deterministisch.

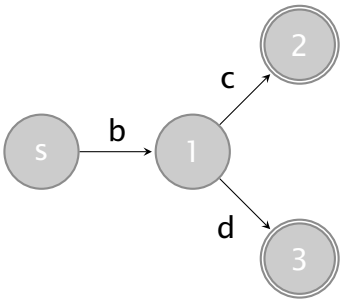


Abbildung 3.13: Zustandsdiagramm eines deterministischen endlichen Automaten (DFA) für das Inhaltsmodell aus Listing 3.19

Ein Beispiel für ein nicht-deterministisches Inhaltsmodell, dass sich nicht in ein 1-nicht-ambiges (deterministisches) umschreiben lässt, wäre das in Listing 3.32 auf Seite 79 dargestellte.

Streng genommen muss unterschieden werden zwischen dem Vorliegen einer eindeutigen (deterministischen) Auswahl von Partikeln (genau diese wird von der UPA forciert) und dem Vorliegen eines vollständig deterministischen Inhaltsmodells (im Sinne eines vollständig deterministischen Automatens). Daher verlangt die UPA nicht zwingend eine Grammatik, die einen eindeutigen Parse-Tree (also eine eindeutige Ableitung) zulässt.

Bei der Betrachtung von Determinismus muss weiter unterschieden werden, ob Zeichenkette (wie bei Brüggemann-Klein und Wood 1997a) oder Bäume der Untersuchungsgegenstand sind. Bei ersteren ist ein deterministisches Inhaltsmodell 1-nicht-ambig (sofern die Zeichenketten von links nach rechts traversiert werden). Bei Bäumen kann

keine allgemeine Aussage getroffen werden, da Determinismus unter anderem abhängig ist von der Strategie, mit der der Baum verarbeitet wird (Top-Down vs. Bottom-Up, etc.).

Interessant in diesem Zusammenhang ist die Tatsache, dass mehrfache Vorkommen eines Kindelements in einem Inhaltsmodell zwar prinzipiell durch die Okkurrenzindikatoren ? (fakultativ), + (mindestens einmalig) und * (beliebig oft) aber auch durch alternative Konstrukte, wie das in Listing 3.20 gezeigte, ausgedrückt werden können (Goldfarb und Prescod 2004, S. 806f.).

Listing 3.20: Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante A)

```
1 <!ELEMENT a (b, b, b)>
```

Dieses Inhaltsmodell wird auch weiterhin als deterministisch angesehen. Interessanterweise wird auch das folgende Beispiel (Listing 3.21) von Parsern akzeptiert, wobei beim Auftreten von genau vier Vorkommen des Elements b eine solche eindeutige Vorhersage nicht mehr möglich sein sollte:

Listing 3.21: Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante B)

```
1 <!ELEMENT a (b, b, b, b?, b?)>
```

Anmerkung Es muss festgehalten werden, dass die Forderung deterministischer Inhaltsmodelle nur aufgrund der Kompatibilität zu SGML Eingang in den *nicht-normativen* Teil der XML Spezifikation gefunden hat. Das Fehlen eines solchen wird von der verarbeitenden Software als Fehler gewertet: „XML processors built using SGML systems may flag non-deterministic content models as errors.“ (Bray, Paoli, Sperberg-McQueen, Maler und Yergeau 2008, Abschnitt E, „Deterministic Content Models“). Ein Fehler ist zu unterscheiden von einem schwer wiegenden Fehler (*fatal error*), der in jedem Fall zu einem Abbruch der Verarbeitung führt. Bei Auftreten eines (einfachen) Fehlers dagegen kann der Parser die Verarbeitung fortführen, eine Warnung ausgeben oder die Verarbeitung abbrechen (Graham und Quin 1999, S. 279).

Die Umwandlung eines nicht-deterministischen Inhaltsmodells in ein deterministisches analog zu der im Listing 3.19 auf der vorherigen Seite wird in den meisten Fällen durch den Parser transparent durchgeführt (Graham und Quin 1999, S. 280). Ein von H. Chen und P. Lu (2011) vorgeschlagener Algorithmus zur besseren Hilfestellung bei der Erstellung deterministischer Ausdrücke (und damit Inhaltsmodelle) könnte diese Arbeit weiter verbessern.

XSD erlaubt eine solche Konstruktion nicht; die in Listing 3.22 auf der nächsten Seite aufgeführte Elementdeklaration wird vom Parser zurückgewiesen. Hier greift die bereits erwähnte und in XML SCHEMA explizit spezifizierte Unique Particle Attribution. Die UPA verbietet damit explizit einen Look-Ahead im Sinne der von Brüggemann-Klein und Wood (1997a) diskutierten *1-unambiguity* und erlaubt damit, Elemente einer XML-Instanz eindeutig einem Partikel in einem Inhaltsmodell zuzuordnen (vgl. auch Abschnitt 3.4.4.4 und van der Vlist 2003c, S. 104ff.).

Im Gegensatz zu DTD werden Elementdeklarationen mit potentiell ambigen Inhaltsmodell von XML SCHEMA unterstützenden Parsern strikt zurückgewiesen.[44] Zulässig

[44] Für eine detaillierte Betrachtung der UPA inklusive eines Algorithmus' vgl. Fuchs und A. Brown (2003).

Listing 3.22: Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante B/XSD)

```
<xs:element name="a">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="b"/>
   <xs:element ref="b"/>
   <xs:element ref="b"/>
   <xs:element ref="b" minOccurs="0" maxOccurs="1"/>
   <xs:element ref="b" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

ist in XSD dagegen wiederum die sehr nahe Umschreibung aus Listing 3.23 mit der DTD-Entsprechung in Listing 3.24.

Listing 3.23: Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante B 2/XSD)

```
<xs:element name="a">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="b"/>
   <xs:element ref="b"/>
   <xs:element ref="b"/>
   <xs:sequence minOccurs="0">
    <xs:element ref="b"/>
    <xs:element ref="b" minOccurs="0"/>
   </xs:sequence>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Listing 3.24: Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante B 2)

```
<!ELEMENT a (b, b, b, (b, b?)?)>
```

Anmerkung Natürlich sieht XML SCHEMA für die Spezifikation eines mehrfach vorkommenden Kindelements innerhalb eines Inhaltsmodells die Nutzung der Attribute `minOccurs` und `maxOccurs` vor, die wesentlich einfacher ein Inhaltsmodell der gewünschten Art erzeugen lassen. Im aktuellen Dikussionskontext geht es aber darum aufzuzeigen, wo die Grenzen der beiden Grammatikformalismen in Bezug auf deterministische Inhaltsmodelle liegen bzw. inwieweit Determinismus ein wesentlicher Bestandteil darstellt.

RELAX NG dagegen erlaubt auch nicht-deterministische Inhaltsmodelle – und kennt auch keine andere als die in den Listings 3.25 und 3.26 (erstere in der XML-Syntax, letztere in der *Compact Syntax*) gezeigte Möglichkeit, um ein Vorkommen von minimal drei und maximal fünf Kindelementen festzulegen (van der Vlist 2003b, S. 21f.).

Bei gemischten Inhaltsmodellen (also dem Auftreten von Zeichenketten und Elementen) dagegen sind in DTD explizite mehrfache Vorkommen des gleichen Elementtyps

Listing 3.25: Inhaltsmodell mit mehrfach vorkommendem Kindelement (RELAX NG/XML Syntax)

```
<element name="a">
 <element name="b">
  <text/>
 </element>
 <element name="b">
  <text/>
 </element>
 <element name="b">
  <text/>
 </element>
 <optional>
  <element name="b">
   <text/>
  </element>
 </optional>
 <optional>
  <element name="b">
   <text/>
  </element>
 </optional>
</element>
```

Listing 3.26: Inhaltsmodell mit mehrfach vorkommendem Kindelement (RELAX NG/Compact Syntax)

```
element a {
  element b { text },
  element b { text },
  element b { text },
  element b { text }?,
  element b { text }?
}
```

verboten (vgl. Bray, Paoli, Sperberg-McQueen, Maler und Yergeau 2008, Abschnitt 3.2.2, „Mixed Content“ und Listing 3.27).

Listing 3.27: Gemischtes Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante C)

```
<!ELEMENT a (#PCDATA | b, b)*>
```

Mit einer solchen Konstruktion gibt es in XML SCHEMA wiederum keine Probleme (vgl. Listing 3.28 auf der nächsten Seite). Allerdings greift auch hier die UPA, sofern das Inhaltsmodell ambig wird (vgl. Listing 3.29 auf der nächsten Seite).

Wie van der Vlist (2003c, S. 108) ausführt, liegt die Motivation, das Verbot ambiger Inhaltsmodelle in XML SCHEMA nicht nur beizubehalten, sondern es mit der UPA sogar noch in den normativen Teil der Spezifikation aufzunehmen, darin begründet, „einfach zu implementierende Schema-Prozessoren zu ermöglichen, beispielsweise auf der Grundlage endlicher Automaten (FSM). Die Ausführungszeit [Anm.: genauer wäre ‚Zustandsmenge‘] solcher Automaten kann exponentiell wachsen, wenn die UPA-Regel

Listing 3.28: Gemischtes Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante C/XSD)

```
<xs:element name="a">
 <xs:complexType mixed="true">
  <xs:sequence>
   <xs:element ref="b"/>
   <xs:element ref="b"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Listing 3.29: Gemischtes Inhaltsmodell mit mehrfach vorkommendem Kindelement (Variante D/XSD)

```
<xs:element name="a">
 <xs:complexType mixed="true">
  <xs:sequence>
   <xs:element ref="b" minOccurs="0"/>
   <xs:element ref="b" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

verletzt wird."[45] Neben den FSM nennt van der Vlist (2007, S. 42) auch die *Brzozowski-Ableitung* (*Brzozowski Derivatives*, vgl. Brzozowski 1964) als zweiten hauptsächlich genutzten Algorithmus zur Verarbeitung von Grammatik-basierten Schemasprachen. Der Einsatz von Brzozowski-Ableitungen zur Verarbeitung von Grammatiken auf Basis von XSD wird auch von Sperberg-McQueen (2005) erörtert. Tennison (2007, S. 15f.) demonstriert den Einsatz von Brzozowski-Ableitungen zur Validierung der experimentellen Constraint Language CREOLE. Dal Zilio und Lugiez (2003) diskutieren mit den *Sheaves Automata* erweiterte Baumautomaten als weiteren Algorithmus, der sich besonders für die Verarbeitung von XML-Schemata eignet.

Ebenfalls der vereinfachten Überprüfung der Inhaltsmodelle zum Opfer gefallen ist der in SGML noch vorhandene und aus XML entfernte Interleave-Konnektor (AND bzw. &). Argumentiert wird damit, dass sich Inhaltsmodelle entsprechend umschreiben lassen (vgl. Listings 3.30 und 3.31).

Listing 3.30: Interleave-Operator in SGML

```
<!ELEMENT a - - (b & c)>
```

Die entsprechende Umformulierung in XML:

Listing 3.31: Entsprechende Umformulierung des Interleave-Operators in XML-DTD

```
<!ELEMENT a ((b,c) | (c,b))>
```

[45] Eventuelle Probleme mit nicht-deterministischen Inhaltsmodellen lassen sich vermeiden, indem ein deterministischer FSM (DFA, *Deterministic Finite State Automata*) durch Ableitung eines nicht-deterministischen FSM (NFA, *Non-deterministic Finite State Automata*) erzeugt wird – wobei die Gefahr besteht, eine exponentielle Menge an Zuständen zu benötigen. Wie bereits Rabin und Scott (1959) zeigen, lassen sich NFA in DFA überführen – allerdings unter der Gefahr, dass die Anzahl der dafür nötigen Zustände exponentiell sein kann.

Wie bereits in der SGML-Spezifikation ausgeführt, kann die hier angesprochen Reduzierung („reduction of *and groups*“) zu einer sehr hohen Anzahl an resultierenden Sequenz-Inhaltsmodellen führen („*seq groups*“); bei sechs Elementen in der ursprünglichen Gruppe zu 6! = 720 Gruppen (Goldfarb 1991b, S. 559). Kilpeläinen (1999) diskutiert eine Reihe von Ansätzen zur Umformulierung von SGML *and groups* in entsprechende *seq groups* (um SGML-Inhaltsmodelle in XML-Inhaltsmodelle zu transformieren). Dabei können je nach verwendetem Ansatz auch nicht-deterministische Model Groups entstehen. Das in Listing 3.32 gezeigte Inhaltsmodell (nach Kilpeläinen 1999, S. 4) des Elements c wird je nach Parser akzeptiert, während die Variante aus Listing 3.33 aufgrund der strikteren UPA zurückgewiesen wird.

Listing 3.32: Interleave-Beispiel in XML-DTD

```
<!ELEMENT c ((a,b?)|(b?,a))>
<!ELEMENT a EMPTY>
<!ELEMENT b EMPTY>
```

Listing 3.33: Interleave-Beispiel in XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <xs:element name="c">
  <xs:complexType>
   <xs:choice>
    <xs:sequence>
     <xs:element ref="a"/>
     <xs:element minOccurs="0" ref="b"/>
    </xs:sequence>
    <xs:sequence>
     <xs:element minOccurs="0" ref="b"/>
     <xs:element ref="a"/>
    </xs:sequence>
   </xs:choice>
  </xs:complexType>
 </xs:element>
 <xs:element name="a"/>
 <xs:element name="b"/>
</xs:schema>
```

Die von Kilpeläinen (1999) favorisierte Umwandlungsstrategie ermöglicht nicht nur deterministische Inhaltsmodelle, sondern hält auch die Anzahl der resultierenden *seq groups* unterhalb der im SGML-Standard aufgeführten $n!$ (für n Mitglieder einer *and group*).

Weitere Algorithmen sind in der Lage deterministische von nicht-deterministischen Inhaltsmodellen zu unterscheiden (vgl. Brüggemann-Klein 1993b, für eine Überprüfung in linearer Zeit; oder auch Kawaguchi 2001a; sowie Kilpeläinen 2011) oder ermöglichen auch beim Vorliegen eines nicht-deterministischen Inhaltsmodells eine lineare Verarbeitungszeit (vgl. Murata, D. Lee, Mani und Kawaguchi 2005; Hosoya 2010). Auch stehen seit Veröffentlichung von RELAX NG entsprechende Validierungsalgorithmen zur Verfügung (z. B. Clark 2002; *Hedge Automata*, vgl. Murata 1999; Brüggemann-Klein und Wood 2004; Murata, D. Lee, Mani und Kawaguchi 2005).

3.4.4 Formale Ausdrucksstärke

Die verschiedenen Formalismen zur Beschreibung von XML Dokumentgrammatiken unterscheiden sich nicht nur in Bezug auf ihre Syntax, vielmehr sind sie auch unterschiedlich mächtig in Hinblick auf ihre Ausdrucksstärke. Eine große Ausdrucksstärke (vom Standpunkt der formalen Sprachen her gesehen) hat nicht nur Vorteile: Je mächtiger ein Grammatikformalismus ist, umso mehr Ressourcen werden prinzipiell für die Verarbeitung damit erstellter Grammatiken benötigt. So erweitern XML SCHEMA und RELAX NG die Möglichkeiten von XML-DTD: XML SCHEMA führt eine ganze Reihe an vordefinierten Datentypen ein und erlaubt die Definition eigener. Darüber hinaus wird unterschieden zwischen lokal und global deklarierten Einheiten der Informationsmodellierung und XML NAMESPACES werden unterstützt.[46] RELAX NG erlaubt dagegen nicht-deterministische Inhaltsmodelle (van der Vlist 2003b, S. 316) und führt den Interleave-Operator (analog zum Konnektor & in SGML, vgl. Goldfarb 1991b, S. 291) wieder ein.[47]

Die formale Betrachtung von Grammatikformalismen hat bereits eine lange Tradition: Die Ursprünge lassen sich bis in das Jahr 1955 zurückverfolgen; in diesem Jahr veröffentlichte Chomsky seine Theorie der formalen Grammatik (Chomsky 1955; Chomsky 1956), worauf 1959 die Chomsky-Hierarchie folgte (Chomsky 1959).

Als Beispiel für die folgende Betrachtung sei eine Dokumentgrammatik für einen Text gegeben, der einen optionalen Titel, gefolgt von mindestens einem Abschnitt (alternativ einem Absatz) enthält. Informationen über einen Autoren können ebenfalls angegeben werden. Abschnitte können optional eine Angabe zum Typ haben, Absätze einen eindeutigen Bezeichner. Der Titel besteht aus einer Zeichenkette, Absätze aus Zeichenketten oder einem Verweis, der wiederum bis auf die Referenz auf das Verweisziel leer ist.

Ohne Nutzung eines bestimmten Grammatikformalismus könnte die Dokumentgrammatik wie folgt ausgedrückt werden:

$$
\begin{aligned}
&S \rightarrow text(author?, Title?(Section|Para))\\
&Section \rightarrow section(type?, Title?(Section|Para))\\
&Title \rightarrow title(\#pcdata)\\
&Para \rightarrow para(id?, \#pcdata|Xref)\\
&Xref \rightarrow xref(href, \epsilon)
\end{aligned}
$$

Zur Notation In diesem und den folgenden Abschnitten werden einige Aspekte mittels formaler Notation erläutert, die an dieser Stelle kurz eingeführt wird.

- Nichtterminale in Produktionsregeln beginnen mit einem Großbuchstaben.

[46] DTD kennt nur die globale Deklaration von Elementen und die lokale Deklaration von Attributen. Dieser Umstand und damit das Fehlen einer Möglichkeit, eine gewisse Kontextsensitivität auszudrücken, wurde bereits in Welty und Ide (1999) bemängelt – Sperberg-McQueen (2000) zeigt Lösungsbeispiele für kontextsensitive Regeln in XML SCHEMA.

[47] Im Gegensatz zum Konnektor & erlaubt der Interleave-Operator in RELAX NG zusätzlich noch, dass Mitglieder zweier Gruppen sich untereinander vermischen lassen. Van der Vlist (2003b, S. 54f.) demonstriert dies an einem Beispiel eines Museums, in dem sowohl Besuchergruppen als auch einzelne Besucher die Ausstellung besuchen. Die einzelnen Besucher können sowohl vor oder nach einer Gruppe das Museum durchwandern, als auch sich unter die Gruppenmitglieder mischen.

- Knotenlabel/Terminale beginnen mit einem Kleinbuchstaben.
- Produktionsregeln sind in der Form $A \rightarrow a(r)$ notiert, wobei im Beispiel A die linke Seite einer Regel (Nichtterminal) repräsentiert, a das Terminal/Knotenlabel, das durch die Regel eingeführt wird, und r das entsprechende Inhaltsmodell.
- $X\_\_$ und $\_\_X$ bezeichnen den linken bzw. rechten Geschwisterknoten X.
- Die Sprache, die aus einem Nichtterminal A generiert wird, wird mit $L(A)$ denotiert.

Im Beispiel sind Attribute und Zeichenketten als Terminalsymbole definiert. Es sei erneut darauf hingewiesen, dass diese Grammatik Bäume und nicht Zeichenketten generiert. Die Nichtterminale verbleiben nicht als Label der Knoten im Baum, die sie einführen, sondern werden durch die Terminallabel ersetzt.

Eine Beispiel-XML-Instanz ist in Listing 3.34 aufgeführt.

Listing 3.34: Eine Beispiel-XML-Instanz

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <text author="maik">
3   <title>A simple title</title>
4   <section>
5    <title>A section title</title>
6    <para id="p1">Introductory para</para>
7    <section>
8     <title>A subsection title</title>
9     <para>Some text with a reference: <xref href="p1"/>.</para>
10   </section>
11  </section>
12 </text>
```

3.4.4.1 Reguläre Baumgrammatiken

Der allgemeinste Fall einer Baumgrammatik, der im Rahmen dieser Arbeit untersucht wird, ist die reguläre Baumgrammatik. Murata, D. Lee, Mani und Kawaguchi (2005) folgend, lässt sich eine reguläre Baumgrammatik (*Regular Tree Grammar, RTG*) wie folgt definieren:

Definition 2 *Eine reguläre Baumgrammatik (*Regular Tree Grammar, RTG*) ist ein 4-Tupel (N,T,S,P), wobei gilt:*

- *N ist eine endliche Menge von Nichtterminalen,*
- *T ist eine endliche Menge von Terminalen,*
- *S ist das Startsymbol, das in N enthalten ist,*
- *P ist eine Menge von Produktionsregeln in der Form: $A \rightarrow a(r)$, wobei gilt: $A \in N$, $a \in T$, und r ist ein regulärer Ausdruck über Elemente aus N. A repräsentiert die linke Seite einer Regel, a das Terminal oder auch Label, das durch die Regel eingeführt wird und r das entsprechende Inhaltsmodell.*

Die Klasse von Sprachen, die von einer solchen RTG erzeugt werden, wird die Klasse der regulären Baumsprachen (*Regular Tree Languages, RTL*s) genannt und ist im Rahmen dieser Arbeit die allgemeinste Klasse. Die von Murata, D. Lee, Mani und Kawaguchi (2005) eingeführten restringierteren Klassen lassen sich von den RTLs ableiten.

Definition 3 *Zwei Regeln einer RTG werden konkurrierend genannt, wenn sie die gleichen terminalen Knoten einführen, aber unterschiedliche linke Seiten haben:* $A \to a(r)$ *und* $B \to a(r')$ *sind konkurrierende Regeln.*

Prinzipiell lassen sich in einer RTG zwei Regeln mit der gleichen linken Seite, die das gleiche Terminal einführen, kombinieren, indem die jeweiligen Inhaltsmodelle zusammengeführt werden. Das gilt, da sich für jegliche zwei regulären Ausdrücke ein einfacher regulärer Ausdruck formulieren lässt, der die Vereinigung der beiden vorherigen denotiert.

Aus diesem Grund wird in dieser Arbeit von der Annahme ausgegangen, dass in den untersuchten Grammatiken keine zwei Regeln mit der gleichen linken Seite und dem gleichen Terminal vorkommen. Daraus lässt sich weiter folgern, dass das Konzept der konkurrierenden Regeln ein wesentlicher Bestandteil bei der Behandlung von Determinismus und Ambiguität ist und dass man fast ebenso gut von konkurrierenden Nichtterminalen sprechen kann: Konkurrierende Nichtterminale stehen auf der linken Seite zweier konkurrierender Regeln; eine Grammatik hat immer genau so viele konkurrierende Regeln wie sie konkurrierende Nichtterminale besitzt.

Weitere Grammatiktypen werden im Folgenden analog zu Murata, D. Lee, Mani und Kawaguchi (2005) definiert.

3.4.4.2 Lokale Baumgrammatiken

Definition 4 *Eine lokale Baumgrammatik (*Local Tree Grammar, LTG*) ist eine reguläre Baumgrammatik, die keine konkurrierende Regeln enthält.*

In einer LTG gibt es eine 1:1-Beziehung zwischen Nichtterminalen und Terminalen, weshalb sie einer CFG sehr ähnlich ist (allerdings nicht identisch, da reguläre Inhaltsmodelle beliebig viele Verzweigungen erlauben).

Ein Beispiel für eine lokale Baumgrammatik ist die in Listing 3.35 gezeigte DTD.

Listing 3.35: XML-DTD als Vertreter einer lokalen Baumgrammatik

```
1 <!ELEMENT text (title?, (section | para)+)>
2 <!ATTLIST text author CDATA #IMPLIED>
3 <!ELEMENT title (#PCDATA)>
4 <!ELEMENT section (title?, (section | para)+)>
5 <!ATTLIST section type (global | sub) #IMPLIED>
6 <!ELEMENT para (#PCDATA | xref)*>
7 <!ATTLIST para id ID #IMPLIED>
8 <!ELEMENT xref EMPTY>
9 <!ATTLIST xref href IDREF #REQUIRED>
```

Wie oben angegeben, unterstützt DTD nur global deklarierte Elemente (und lokal deklarierte Attribute). Darüber hinaus sind konkurrierende Regeln, also Regeln, die den

gleichen terminalen Knoten einführen, also ein Element gleichen Namens (bzw. mit gleichem *local name*), verboten. Konkret bedeutet das für das Beispiel, dass die Inhaltsmodelle der Elemente `text` und `section` die gleichen drei Elemente `title`, `section` und `para` referenzieren. Als besondere Eigenschaft von DTD gilt die Behandlung von *mixed content*, also einem Inhaltsmodell, das sowohl aus Zeichenketten als auch Elementen besteht (am Beispiel des Elements `para`): Hier ist zwingend als Konnektor das logische Oder „|“ und der Stern „*“ als Okkurrenzindikator (beliebig oft) vorgesehen. Da DTD den Integritätsmechanismus per `ID`-/`IDREF`/`IDREFS`-Attribut unterstützt, wird dieser im Beispiel zur Etablierung von Verknüpfungen zwischen Absätzen (über deren Attribut `id`) und Verweisen (mit Hilfe des leeren Elements `xref`) eingesetzt.

Zu beachten ist, dass DTD auch keinerlei Typisierung von Datentypen vorsieht. Darauf aufbauende Arbeiten wie Buck *et al.* (2000); Papakonstantinou und Vianu (2000); Balmin *et al.* (2004); Martens, Neven, Schwentick und Bex (2006) schlagen die Erweiterung durch Datentypen vor, DTD++, (Vitali *et al.* 2003) fügt zusätzlich die Unterstützung von XML NAMESPACES hinzu, während DTD++ 2.0, (Fiorello *et al.* 2004) darüber hinaus *Co-Constraints* (vgl. Abschnitt 3.4.4.4) erlaubt.

3.4.4.3 Single Type Tree Grammar

Definition 5 *Eine* Single Type Tree Grammar (STG) *ist eine reguläre Baumgrammatik, in der konkurrierende Nichtterminale nicht im gleichen Inhaltsmodell auftauchen dürfen.*

STG schließen im Gegensatz zu LTG konkurrierende Nichtterminale nicht generell aus, sondern verbieten sie nur im gleichen Inhaltsmodell. Zu beachten ist wieder, dass Nichtterminale in einer XML-Grammatik durch das korrespondierende Terminal ersetzt werden, d. h., in einer konkreten XML-Grammatik spricht man bei dieser Einschränkung von dem zu deklarierenden Element. Oder anders ausgedrückt: Ein einer STG entsprechendem XML-Grammatikformalismus erlaubt die Definition von Grammatiken, die Deklarationen des gleichen Elementnamens (mit durchaus unterschiedlichem Inhaltsmodell) beinhalten, sofern sie nicht im gleichen Inhaltsmodell (im gleichen Kontext) auftauchen. Dieses Verbot konkurrierender Nichtterminale im gleichen Inhaltsmodell wird in der XML SCHEMA-Spezifikation (Thompson *et al.* 2004, Abschnitt 3.8.6, „Constraints on Model Group Schema Components“) als *Element Declarations Consistent (EDC) Constraint* bezeichnet. Damit erlaubt diese Definition die Verwendung lokal deklarierter Elemente und beschreibt ein Merkmal von XML SCHEMA. Murata, D. Lee, Mani und Kawaguchi (2005, S. 667) folgend, lässt sich XML SCHEMA als *Single Type Tree Grammar* klassifizieren, d. h. für jedes Knotenlabel im Baum kann eine eindeutige Interpretation bestimmt werden, sofern die Interpretation des Mutterknotens (und damit des Kontexts) bekannt ist.[48] Die Einbeziehung der Mutterknoten ist notwendig, um konkurrierende Regeln zu berücksichtigen. Konkurrierende Regeln (genauer: konkurrierende Nichtterminale) sind wie folgt aufgebaut: Auf der linken Seite einer Produktionsregel stehen die

[48] Der Terminus *Single Type* lässt sich ableiten aus Fallside und Walmsley (2004, Abschnitt 4.5, „Redefining Types & Groups“): „It is illegal to have two elements of the same name (and in the same target namespace) but different types in a content model [...]“.

unterschiedlichen Nichtterminale *A* und *B*, die auf der rechten Seite der Regel beide das gleiche Terminal haben.

Listing 3.36: Realisierung der Beispielgrammatik mittels XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="text">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="textType">
     <xs:attribute name="author" type="xs:string" use="optional"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="title" type="xs:string"/>
 <xs:element name="section">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="textType">
     <xs:attribute name="type">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="global"/>
        <xs:enumeration value="sub"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:attribute>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="para">
  <xs:complexType mixed="true">
   <xs:sequence>
    <xs:element name="xref" minOccurs="0">
     <xs:complexType>
      <xs:attribute name="href" type="xs:IDREF" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute ref="id" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:attribute name="id" type="xs:ID"/>
 <xs:complexType name="textType">
  <xs:sequence>
   <xs:element ref="title" minOccurs="0"/>
   <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:group name="sectOrPara">
  <xs:choice>
   <xs:element ref="section"/>
   <xs:element ref="para"/>
  </xs:choice>
 </xs:group>
</xs:schema>
```

In der in Listing 3.36 dargestellten Implementierung sind keine lokal deklarierten Elemente gleichen Namens aufgeführt, da es in der Beispielgrammatik keine Anhaltspunkte gibt. Zwar wäre es möglich, das Element `title` mehrfach zu deklarieren (je

nachdem, ob es Kind des Elements `text` oder des Elements `section` ist), da die Strukturierung eines Titels umgangssprachlich aber in beiden Kontexten identisch ist, wäre das ein sehr konstruiertes Beispiel. Denkbar wären aber andere Grammatiken, die z. B. ein Element `name` je nach Kontext unterschiedlich deklarieren: Als Kind eines Elements `person` mit den Kindelementen `surname` und `lastName` und als Kind eines Elements `product` als reines Datenelement.[49]

Das XML-Schema in Listing 3.36 auf der vorherigen Seite ist nur eine mögliche Implementierung der Beispielgrammatik und ist so konzipiert, dass es einige der Merkmale von XML SCHEMA aufzeigt, die diesen Grammatikformalismus von DTD unterscheiden: Das Element `text` wird deklariert mit Hilfe einer Ableitung durch Erweiterung des global deklarierten Typs `textType`, der sich wiederum auf die global deklarierte Model Group `sectOrPara` bezieht. Es gibt sowohl lokal als auch global deklarierte Attribute (vgl. `author` und `id`) und Elemente (vgl. `xref` mit den anderen Elementdeklarationen). Damit sind prinzipiell deutlich mehr Möglichkeiten zur Strukturierung einer Auszeichnungssprache gegeben, als DTD es ermöglicht. Wie allerdings bereits Martens, Neven, Schwentick und Bex (2006, S. 771, 786) ausführen, können aufgrund des in Definition 5 auf Seite 83 formal ausformulierten *Element Declarations Consistent (EDC) Constraint* reguläre Baumsprachen (*Regular Tree Languages*) mit XML SCHEMA nicht modelliert werden. Damit folgen Martens, Neven, Schwentick und Bex der in Murata, D. Lee, Mani und Kawaguchi (2005) geäußerten Klassifizierung XML SCHEMAS als *Single Type Tree Grammar*.

Zu beachten ist allerdings eine Einschränkung: Bei Nutzung der in XML SCHEMA (und auch in RELAX NG) vorhandenen *Wildcards* – Platzhaltern, die Elemente oder Attribute an ihrer Stelle zulassen, ohne sie hinsichtlich des Namens einzuschränken –, ist das entsprechende Schema nicht mehr eine *Single Type Tree Grammar*, sondern eine *Restrained-Competition Tree Grammar* (vgl. Murata, D. Lee, Mani und Kawaguchi 2005, S. 675f. und Abschnitt 3.4.4.5). XML SCHEMA stellt für die Verwendung von Wildcards die Elemente `xs:any` und `xs:anyAttribute` zur Verfügung. Eine mögliche Erweiterung der Beispielgrammatik könnte – in Anlehnung an Murata, D. Lee, Mani und Kawaguchi (2005, S. 676) – aussehen wie in Listing 3.37.

Listing 3.37: Element-Wildcards in XML Schema

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="text">
4    <xs:complexType>
5     <xs:complexContent>
6      <xs:extension base="textType">
7       <xs:attribute name="author" type="xs:string" use="optional"/>
8      </xs:extension>
9     </xs:complexContent>
10   </xs:complexType>
11  </xs:element>
12  <xs:element name="section">
13   <xs:complexType>
```

[49] Für eine solche Unterscheidung sind streng genommen keine zwei lokal deklarierten Elemente gleichen Namens notwendig, es wäre ebenso möglich, ein einzelnes Element `name` zu deklarieren, dessen Inhaltsmodell aus der Abfolge `surname`, `lastName` oder alternativ einer reinen Zeichenkette besteht. Das Beispiel soll daher nur der Illustration dienen.

```
14    <xs:complexContent>
15     <xs:extension base="textType">
16      <xs:attribute name="type">
17       <xs:simpleType>
18        <xs:restriction base="xs:string">
19         <xs:enumeration value="global"/>
20         <xs:enumeration value="sub"/>
21        </xs:restriction>
22       </xs:simpleType>
23      </xs:attribute>
24     </xs:extension>
25    </xs:complexContent>
26   </xs:complexType>
27  </xs:element>
28  <xs:element name="para">
29   <xs:complexType mixed="true">
30    <xs:sequence>
31     <xs:element name="xref" minOccurs="0">
32      <xs:complexType>
33       <xs:attribute name="href" type="xs:IDREF" use="required"/>
34      </xs:complexType>
35     </xs:element>
36    </xs:sequence>
37    <xs:attribute ref="id" use="optional"/>
38   </xs:complexType>
39  </xs:element>
40  <xs:element name="title" type="xs:integer"/>
41  <xs:attribute name="id" type="xs:ID"/>
42  <xs:complexType name="textType">
43   <xs:sequence>
44    <xs:element name="title" type="xs:string"/>
45    <xs:any/>
46    <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
47   </xs:sequence>
48  </xs:complexType>
49  <xs:group name="sectOrPara">
50   <xs:choice>
51    <xs:element ref="section"/>
52    <xs:element ref="para"/>
53   </xs:choice>
54  </xs:group>
55 </xs:schema>
```

Der complexType `textType` wurde geändert, so dass es neben dem jetzt lokal deklarierten Element `title` eine Element-Wildcard in Form des Elements `xs:any` (Zeile 45) hinzugefügt wurde. Zusätzlich wurde ein neues global deklariertes Element `title` vom Typ `xs:integer` erstellt. In der Instanz muss unterhalb der Elemente `text` bzw. `section` das Kindelement `title` zweimal vorkommen: Einmal mit dem Datentyp `xs:string`, das darauf folgende mit dem Datentyp `xs:integer` (vgl. Listing 3.38).

Listing 3.38: Element-Wildcards in der XML-Instanz

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <text xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="text_any.xsd" author="maik">
4   <title>A simple title</title>
5   <title>1</title>
6   <section>
7    <title>A section title</title>
8    <para id="p1">Introductory para</para>
9    <section>
10    <title>A subsection title</title>
```

```
    <title>2</title>
    <para>Some text with a reference: <xref href="p1"/>.</para>
   </section>
  </section>
</text>
```

Zu beachten ist, dass bei lokal deklariertem `title`-Element keine anderen Werte für die Okkurrenz-Indikator-Attribute `minOccurs` und `maxOccurs` vergeben werden dürfen als der Standardwert *1*, da ansonsten erneut die UPA verletzt werden würde.[50] Gleiches gilt für das Element `xs:any` im Beispiel. Die in Listing 3.37 auf Seite 85 gezeigte Grammatik unterscheidet sich daher von den bisher verwendeten derart, dass das Kindelement `title` nicht mehr optional ist (und in diesem Fall sogar zweimal in der Instanz vorkommen muss). Ein reales Anwendungsbeispiel für die Verwendung von Element-Wildcards ist auch in der Diskussion von XSTANDOFF (vgl. Teil III) zu sehen.

Attribut-Wildcards können dazu genutzt werden, um mittels XML-Instanzen Graphen bzw. eXtended Trees (vgl. Abschnitt 3.3.3) zu kodieren. Listing 3.39 zeigt eine modifizierte Fassung des XML-Schemas aus Listing 3.36 auf Seite 84: Es wurden zwei neue Attribute `id` und `idrefs` global deklariert, die dann aus allen Elementen per `xs:anyAttribute` referenziert werden können (zu diesem Zweck musste das ehemals als `simpleType` deklarierte Element `title` entsprechend geändert werden).

Listing 3.39: Attribut-Wildcards zur Kodierung von eXtended Trees

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="text">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="textType">
     <xs:attribute name="author" type="xs:string" use="optional"/>
     <xs:anyAttribute/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="title">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:anyAttribute/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="section">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="textType">
     <xs:attribute name="type">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="global"/>
        <xs:enumeration value="sub"/>
       </xs:restriction>
      </xs:simpleType>
```

[50] In den meisten dokumentierten Fällen bezüglich Einschränkungen durch die UPA liegt tatsächlich eine falsche Verwendung der Wildcard-Mechanismen vor, vgl. Obasanjo (2002).

```
33      </xs:attribute>
34      <xs:anyAttribute/>
35     </xs:extension>
36    </xs:complexContent>
37   </xs:complexType>
38  </xs:element>
39  <xs:element name="para">
40   <xs:complexType mixed="true">
41    <xs:sequence>
42     <xs:element name="xref" minOccurs="0">
43      <xs:complexType>
44       <xs:attribute name="href" type="xs:IDREF" use="required"/>
45      </xs:complexType>
46     </xs:element>
47    </xs:sequence>
48    <xs:anyAttribute/>
49   </xs:complexType>
50  </xs:element>
51  <xs:attribute name="id" type="xs:ID"/>
52  <xs:attribute name="idrefs" type="xs:IDREFS"/>
53  <xs:complexType name="textType">
54   <xs:sequence>
55    <xs:element ref="title" minOccurs="0"/>
56    <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
57   </xs:sequence>
58  </xs:complexType>
59  <xs:group name="sectOrPara">
60   <xs:choice>
61    <xs:element ref="section"/>
62    <xs:element ref="para"/>
63   </xs:choice>
64  </xs:group>
65 </xs:schema>
```

Als Folge sind Instanzen wie die in Listing 3.40 möglich. Zu beachten ist, dass das Attribut `idrefs` als vom Datentyp `xs:IDREFS` deklariert ist, d. h., 1:*n*-Beziehungen sind möglich (wie im Beispiel vom Wurzelelement ausgehend). Die Abbildung 3.14 zeigt den entsprechenden eXtended Tree dazu.

Listing 3.40: Attribut-Wildcards in der XML-Instanz

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <text xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="text_any_id.xsd" author="maik" id="root" idrefs="sec1
       sec1sec1">
4   <title id="t1">A simple title</title>
5   <section id="sec1" idrefs="sec1sec1">
6    <title>A section title</title>
7    <para id="sec1p1">Introductory para</para>
8    <section id="sec1sec1">
9     <title>A subsection title</title>
10    <para id="sec1sec1p1" idrefs="root">Some text with a reference: <xref href="sec1sec1p1"/>.</
          para>
11   </section>
12  </section>
13 </text>
```

XML SCHEMA ermöglicht die Ableitung von Typen durch Erweiterung (`xs:extension`, beispielsweise in Listing 3.39) oder Einschränkung (`xs:restriction`, vgl. van der Vlist 2003c, S. 205ff. und S. 223ff.). Eine Besonderheit kommt dabei bei komplexen Typen zum Tragen: Sofern zwei benannte `complexTypes` `A` und `B` definiert sind und `B` von `A`

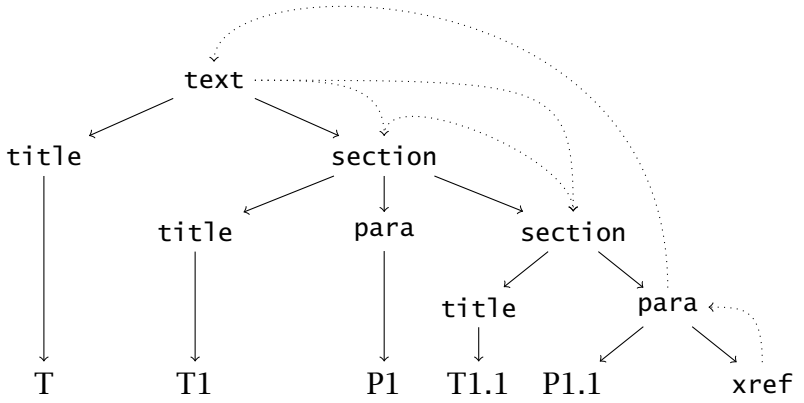


Abbildung 3.14: xT-Darstellung der XML-Instanz aus Listing 3.40

abgeleitet wird, kann ein Element, das im Schema als vom Typ `A` deklariert ist, in der Instanz als vom Typ `B` umdeklariert werden (vgl. van der Vlist 2003c, S. 205ff., Murata, D. Lee, Mani und Kawaguchi 2005, S. 674f.). Murata, D. Lee, Mani und Kawaguchi (2005, S. 206) bilden diese Erweiterung durch das `xsi:type`-Attribut durch zusätzliche Terminalsymbole ab, die den möglichen Werten des Attributs entsprechen. Dadurch behält der Autor der Instanz eine gewisse Flexibilität gegenüber dem Schema-Entwickler.[51] Ein Beispiel für eine solche Typersetzung findet sich in Listing 12.1 in Abschnitt 12.2 dieser Arbeit.

Ein Alleinstellungsmerkmal von XSD ist die Möglichkeit, das Parserverhalten in der Verarbeitung von Teilbäumen zu beeinflussen, die durch `xs:any` bzw. `xs:anyAttribute` eingeführt werden. Mittels des Attributs `processContents` kann in beiden Elementen angegeben werden, ob ein validierender Parser das Element (bzw. den entsprechenden Teilbaum) in der Instanz *immer* auf Validität hin überprüfen soll (Attributwert *strict*, d. h., es *muss* ein entsprechendes Schema vorhanden sein), eine Überprüfung nur bei Vorliegen einer Dokumentgrammatik vornehmen (Attributwert *lax*) oder das Element (den Teilbaum) von der Validierung ausschließen soll (Attributwert *skip* – eine Überprüfung auf Wohlgeformtheit findet weiterhin statt). Diese Möglichkeit der Einflussnahme wird vorrangig dann relevant, wenn verschiedene Auszeichnungssprachen mittels XML NAMESPACES kombiniert werden sollen, vgl. die Ausführungen in Abschnitt 13.4.

RELAX NG wird in Murata, D. Lee, Mani und Kawaguchi (2005) wie seine Vorläufer-Spezifikationen TREX und RELAX CORE als Vertreter einer *Regular Tree Grammar*, *RTG*, klassifiziert. Im Gegensatz zu DTD oder XML SCHEMA basiert es auf der mathematischen Theorie regulärer Ausdrücke und dem Konzept der bereits erwähnten *Hedge Grammars* (vgl. Murata 1999; van der Vlist 2003b; Murata, D. Lee, Mani und Kawaguchi 2005) sowie der Theorie der Hedge Automata (Murata 1999).[52] Als XML-Schemasprache hat es einige Vorteile gegenüber den beiden zuvor genannten Vertretern: Es bietet sowohl

[51] Ein Schema-Entwickler kann allerdings im Gegenzug durch das `block`-Attribut die Typersetzung für Erweiterungen, Einschränkungen oder Ersetzungsgruppen (*Substitution Groups*) verhindern. Alternativ sorgt der Wert *#all* für ein generelles Verbot (van der Vlist 2003c, S. 228f.).

[52] Wie in Murata, D. Lee, Mani und Kawaguchi (2005, S. 689) dargelegt, lassen sich Baumgrammatiken in Hedge Grammar überführen.

eine XML-Syntax als auch eine vereinfachte Compact Syntax (ähnlich der von DTD), und unterstützt neben den auch in DTD und XML SCHEMA vorhandenen Konnektoren Abfolge („,“ bzw. `xs:sequence`) und Auswahl („|“ bzw. `xs:choice`) auch den von SGML bekannten Interleave-Operator – wobei dieser keinerlei Auswirkungen auf die Ausdrucksstärke von RELAX NG hat. Die beiden Listings 3.41 und 3.42 zeigen den Gebrauch sowohl in der XML- als auch in der Compact Syntax.

Listing 3.41: RELAX-Minimalbeispiel mit Interleave in der XML-Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<element xmlns="http://relaxng.org/ns/structure/1.0" name="a">
 <interleave>
  <element name="b">
   <text/>
  </element>
  <element name="c">
   <text/>
  </element>
 </interleave>
</element>
```

Listing 3.42: RELAX-Minimalbeispiel mit Interleave in der Compact Syntax

```
element a {
  element b { text }
  & element c { text }
}
```

DTD lässt bei Vorliegen eines gemischten Inhaltsmodells nur den Konnektor „|“ (XOR, logisches Oder) zu. XML SCHEMA erlaubt zwar neben `xs:choice` auch zusätzlich die Verwendung von `xs:sequence`, das betrifft aber nur die Festlegung der Abfolge der Elemente, die Teil des gemischten Inhaltsmodells sind. In RELAX NG kann eine Reihenfolge für alle Bestandteile eines gemischten Inhaltsmodells vorgegeben werden, da Textknoten wie Elemente und Attribute behandelt werden (vgl. Listing 3.43 und van der Vlist 2003b, S. 57f).[53]

Listing 3.43: Mixed Content mit Abfolge

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="a" xmlns="http://relaxng.org/ns/structure/1.0">
 <group>
  <text/>
  <element name="b">
   <text/>
  </element>
  <element name="c">
   <text/>
  </element>
 </group>
</element>
```

Alternativ kann mittels des Interleave-Operators das von DTD und XSD bekannte Verhalten erzeugt werden (vgl. Listing 3.44).

Listing 3.44: Mixed Content mit Interleave

```
<?xml version="1.0" encoding="UTF-8"?>
```

[53] Als zusätzliche Eigenheit sind Attribute Teil des Inhaltsmodells eines Elements.

```
<element name="person" xmlns="http://relaxng.org/ns/structure/1.0">
 <interleave>
  <text/>
  <element name="name">
   <text/>
  </element>
  <element name="geburtstag">
   <text/>
  </element>
 </interleave>
</element>
```

Ebenso ermöglicht RELAX NG die in Abschnitt 3.4.4.1 angeführte Kombination zweier Regeln mit der gleichen linken Seite, die das gleiche Terminal einführen, durch Zusammenführung der beiden Inhaltsmodelle mittels eines regulären Ausdrucks, der die Vereinigung beider Regeln denotiert:

> RELAX NG is closed under union: for any two RELAX NG schemas, there is a RELAX NG schema for its union. RELAX NG's composability makes the construction of the union trivial: just wrap the two schemas in a choice element. Closure under union implies that the content model of an element can be context dependent. [...] The design of RELAX NG is informed by the theory of finite tree automata; this makes closure possible and ensures that implementations can be efficient, despite the major increase in expressive power. (Clark 2001a, Abschnitt 6, „Closure“)

Die beiden Regeln $E \rightarrow e(A, B)$ und $E \rightarrow e(\#pcdata|C)*$ sind zwei Regeln mit der gleichen linken Seite (dem gleichen Nichtterminal) E, die das Terminal e einführen. Listing 3.45 zeigt das RELAX NG-Äquivalent des vereinigten regulären Ausdrucks.

Listing 3.45: Anwendungsfall von RELAX NG

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="e" xmlns="http://relaxng.org/ns/structure/1.0">
 <zeroOrMore>
  <choice>
   <group>
    <element name="a">
     <text/>
    </element>
    <element name="a">
     <text/>
    </element>
   </group>
   <group>
    <choice>
     <text/>
     <element name="c">
      <text/>
     </element>
    </choice>
   </group>
  </choice>
 </zeroOrMore>
</element>
```

Sowohl in XML SCHEMA als auch in RELAX NG ist es möglich, zwei gleiche Terminale einzuführen, sofern sie (1) nicht global deklariert sind und (2) das gleiche Inhaltsmodell besitzen, wie die Listings 3.46 und 3.47 zeigen.

Listing 3.46: Korrektes XSD mit zwei gleichen Terminalen gleichen Inhaltsmodells

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="a">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="b" type="xs:string"/>
    <xs:element name="b" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Listing 3.47: Korrektes RELAX-NG-Schema mit zwei gleichen Terminalen gleichen Inhaltsmodells

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="a" xmlns="http://relaxng.org/ns/structure/1.0"
 datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
 <group>
  <element name="b">
   <data type="string"/>
  </element>
  <element name="b">
   <data type="string"/>
  </element>
 </group>
</element>
```

Darüber hinaus erlaubt RELAX NG im Gegensatz zu XML SCHEMA auch die Deklaration zweier Elemente gleichen Namens (zwei Terminale) als Geschwister (im selben Kontext), die ein unterschiedliches Inhaltsmodell haben. Listing 3.48 zeigt ein inkorrektes Beispiel-XSD, Listing 3.49 das korrekte Gegenstück in RELAX NG.

Listing 3.48: Inkorrektes XSD mit zwei gleichen Terminalen unterschiedlichen Inhaltsmodells

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="a">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="b" type="xs:string"/>
    <xs:element name="b" type="xs:date"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Listing 3.49: Korrektes RELAX-NG-Schema mit zwei gleichen Terminalen unterschiedlichen Inhaltsmodells

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="a" xmlns="http://relaxng.org/ns/structure/1.0"
 datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
 <group>
  <element name="b">
```

```
6     <data type="string"/>
7    </element>
8    <element name="b">
9     <data type="date"/>
10   </element>
11  </group>
12 </element>
```

Die in Listing 3.49 gezeigte Deklaration zweier Elemente gleichen Namens im selben Kontext mit unterschiedlichen Inhaltsmodellen kann auch zur Erstellung von Co-occurrence Constraints genutzt werden (vgl. die Ausführungen im Abschnitt 3.4.4.4).

Im Gegensatz zu den beiden anderen vorgestellten Grammatikformalismen, unterstützt RELAX NG keine Integritätsmerkmale via `ID/IDREF/ IDREFS` (ebenso wie andere Datentypen außer den beiden Primitiven `string` und `token`, die sich nur in der Verarbeitung von Weißraum-Zeichen unterscheiden, vgl. van der Vlist 2003c, S. 70f.). Da die Spezifikation allerdings das Laden von externen Datentypen-Bibliotheken unterstützt (z. B. die von XML SCHEMA), führt dies in der Realität zu keiner Einschränkung. Zu beachten ist dabei aber, dass bei Verwendung der in Biron und Malhotra 2004 definierten Datentypen `xs:ID/xs:IDREF/xs:IDREFS` der Verknüpfungsmechanismus in RELAX NG nicht unterstützt wird:

> The semantics defined by [W3C XML Schema Datatypes] for the ID, IDREF and IDREFS datatypes are purely lexical and do not include the cross-reference semantics of the corresponding [XML 1.0] datatypes. The cross-reference semantics of these datatypes in XML Schema comes from XML Schema Part 1. Furthermore, the [XML 1.0] cross-reference semantics of these datatypes do not fit into the RELAX NG model of what a datatype is. Therefore, RELAX NG validation will only validate the lexical aspects of these datatypes as defined in [W3C XML Schema Datatypes]. (Clark und Kawaguchi 2001a).

Um dennoch Gebrauch vom XML-inhärenten Verknüpfungsmechanismus machen zu können, besteht die Möglichkeit, den DTD-Kompatibilitätsmodus von RELAX NG einzusetzen (vgl. Clark und Kawaguchi 2001b; sowie die Ausführungen in van der Vlist 2003b, S. 98–103), da dieser die Verwendung der `xs:ID`-/`xs:IDREF`-/`xs:IDREFS`-Token-Typ-Attribute sowie die Unterstützung von Vorgabewerten für Attribute emuliert[54] – allerdings unter der Einschränkung, dass das betroffene Inhaltsmodell auf das Niveau einer DTD, also einer LTG, restringiert wird: „For compatibility with XML DTDs, RELAX NG DTD Compatibility defines an annotation that can be used to specify default attributes values. However, this can only be used for content models that do not go beyond XML DTDs in their use of attributes.“ (Clark 2001a, Abschnitt 8, „Attributes“). Explizit bedeutet das, dass bei Verwendung der DTD-Kompatibilitäts-Bibliothek die Datentypen `ID`, `IDREF` und `IDREFS` nur in Attributen genutzt werden dürfen, und dass Attribute gleichen Namens unterhalb von Elementen gleichen Namens auch mit dem gleichen Typ deklariert werden müssen. Listing 3.50 demonstriert an einem Beispiel aus van der Vlist (2003b, S. 102) ein nicht korrektes RELAX-NG-Schema.

[54] Von Haus aus werden in RELAX NG keine Vorgabewerte unterstützt, während diese in DTD für Attribute, in XSD zusätzlich auch für Elemente möglich sind.

Listing 3.50: Einschränkungen bei Verwendung der DTD-Kompatibilitäts-Bibliothek

```
1  datatypes d = "http://relaxng.org/ns/compatibility/datatypes/1.0"
2
3  element foo {
4    element bar {
5      attribute id { d:ID }
6    },
7    element bar {
8      attribute id { token }
9    }*
10 }
```

Das Listing 3.51 zeigt eine mögliche Repräsentation der Beispielgrammatik mittels der RELAX NG XML-Syntax, inkl. Laden der Datentypen aus XML SCHEMA (Zeile 3), und unter Nutzung des DTD-Kompatibilitätsmodus (bei den Attributen `id` und href, Zeilen 46 und 54). Um die simultane Verwendung beider Datentypen-Bibliotheken zu demonstrieren, wurde dem Element `text` noch ein optionales Attribut `number` vom Typ `xsd:integer` hinzugefügt (Zeile 7).

Listing 3.51: RELAX NG-Realisierung der Beispielgrammatik

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <grammar xmlns="http://relaxng.org/ns/structure/1.0"
3   datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
4   <start>
5    <element name="text">
6     <optional>
7      <attribute name="number">
8       <data type="integer"/>
9      </attribute>
10    </optional>
11    <optional>
12     <attribute name="number"/>
13    </optional>
14    <choice>
15     <optional>
16      <group>
17       <ref name="element.title"/>
18       <ref name="element.section"/>
19      </group>
20     </optional>
21     <optional>
22      <group>
23       <ref name="element.title"/>
24       <ref name="element.para"/>
25      </group>
26     </optional>
27    </choice>
28   </element>
29  </start>
30  <define name="element.title">
31   <element name="title">
32    <text/>
33   </element>
34  </define>
35  <define name="element.section">
36   <element name="section">
37    <optional>
38     <ref name="element.title"/>
39    </optional>
40    <ref name="sectOrPara"/>
41   </element>
```

```
42  </define>
43  <define name="element.para">
44   <element name="para">
45    <optional>
46     <attribute name="id">
47      <data datatypeLibrary="http://relaxng.org/ns/compatibility/datatypes/1.0" type="ID"/>
48     </attribute>
49    </optional>
50    <zeroOrMore>
51     <choice>
52      <text/>
53      <element name="xref">
54       <attribute name="href">
55        <data datatypeLibrary="http://relaxng.org/ns/compatibility/datatypes/1.0" type="IDREF"
              />
56       </attribute>
57       <empty/>
58      </element>
59     </choice>
60    </zeroOrMore>
61   </element>
62  </define>
63  <define name="sectOrPara">
64   <group>
65    <oneOrMore>
66     <choice>
67      <ref name="element.section"/>
68      <ref name="element.para"/>
69     </choice>
70    </oneOrMore>
71   </group>
72  </define>
73 </grammar>
```

Die in den Zeilen 46 und 54 deklarierten Attribute nutzen explizit den DTD-Kompatibilitätsmodus durch das `datatypeLibaray`-Attribut. RELAX NG-Prozessoren, die sowohl diesen als auch die Einbettung der XSD-Datentypen-Bibliothek unterstützen, sollten laut Spezifikation für `ID`-Datentypen immer die Einmaligkeit des Wertes in der Instanz überprüfen (also auch ohne explizites Laden – für die Datentypen `IDREF/IDREFS` gilt das entsprechende Verhalten). Das Beispiel dient hier der Veranschaulichung. Bei anderer Software ist diese zusätzliche Überprüfung optional:

> The lexical space of ID is the same as the lexical space of NCName. As defined by the W3C XML Schema recommendation, there is one constraint added to its value space: there must not be any duplicate values in a document. RELAX NG doesn't allow datatype libraries to perform this type of check. This is a job for the DTD compatibility feature, as you will see at the end of this chapter. Its specification asks RELAX NG processors supporting this feature to enforce ID uniqueness for W3C XML Schema ID datatypes. Other implementations just check its lexical space as a NCName. [...] The lexical space of IDREF is the same as the lexical space of NCName. Just as for ID, W3C XML Schema adds the constraint that it must match an ID defined in the same document. RELAX NG makes this behavior optional for RELAX NG processors supporting the W3C XML Schema type library without supporting the DTD compatibility feature. (Van der Vlist 2003b, S. 83)

Die hier gezeigte Grammatik ist von der Ausdrucksstärke her nicht mächtiger als die beiden anderen gezeigten Dokumentgrammatiken – insofern geht die Verwendung eines formal mächtigeren Grammatikformalismus' nicht automatisch einher mit einem ausdrucksstärkeren Schema. Die in Abbildung 3.34 auf Seite 81 gezeigte XML-Instanz wäre gegen die bisher gezeigten Dokumentgrammatiken valide (mit Ausnahme des XML-Schemas in Listing 3.37, das Wildcards benutzt). Eine Assoziation zwischen der Instanz und der entsprechenden DTD würde über eine *Doctype Declaration* erfolgen, eine Validierung gegen das XML SCHEMA über Hinzufügen des Namensraums für XML Schema-Instanzen `http://www.w3.org/2001/XMLSchema-instance` und des `noNamespaceSchemaLocation`-Attributs im Wurzelelement `text` (vgl. Listing 3.38). RELAX NG legt keine Assoziation zwischen Instanz und Grammatik fest, da es als Teil eines Multi-Part-Standards zu diesen Zweck auf andere Spezifikationen zurückgreift, wie z. B. ISO/IEC 19757-4:2006.

Wildcards werden auch von RELAX NG unterstützt. Mittels des Elements `anyName` („*“ in der Compact Syntax) können beliebige Elemente oder Attribute an der entsprechenden Stelle im Inhaltsmodell zugelassen werden, über die Kindelemente `nsName` und `except` kann die Auswahl auf andere Namensräume eingegrenzt bzw. bestimmte Namensräume ausgeschlossen werden (van der Vlist 2003b, S. 172). Hier unterscheidet sich RELAX NG von XSD, wo entweder die zugelassenen Namensräume aufgeführt werden können (über eine durch Leerzeichen getrennte Liste) oder über die Schlüsselwerte *##any* (beliebige), *##local* (keinem Namensraum angehörende) oder *##other* (beliebige andere als der Zielnamensraum des aktuellen Schemas), nicht aber Namensräume explizit ausgeschlossen werden können (außer durch Auflistung aller unerwünschten).

Zu beachten ist bei der Verwendung von Wildcards zur Zulassung von Elementen (oder Attributen) aus anderen Namensräumen, dass bei `ID`-Attributen diese über alle Elemente gleichen Namens (ohne Berücksichtigung des Namespaces) gleich deklariert sein müssen analog zum in Listing 3.50 gezeigten inkorrekten Schema. Das bedeutet, dass für solche Elemente keine Unterscheidung zwischen globaler und lokaler Deklaration (hinsichtlich des `ID`-Attributs) gemacht wird, was die Nutzung solcher Integritätsmerkmale in RELAX NG etwas verkompliziert (van der Vlist 2003b, S. 177).

3.4.4.4 Co-Constraints

Co-occurrence Constraints oder auch Co-Constraints, wie sie in Pawson *et al.* (2008) genannt werden, erlauben die Festlegung eines Constraints, d. h. die Beschränkung des Inhaltsmodells eines Elements, in Abhängigkeit vom Vorhandensein oder vom Wert einer (oder mehrerer) anderer Komponente(n) der Informationsmodellierung. Diese Constraints können zwischen Einheiten der Informationsmodellierung (Elementen, Attributen, etc.) oder auch zwischen Datenwerten existieren. Je nach involvierten Komponenten kann man noch weiter unterscheiden zwischen Element-zu-Element-, Element-zu-Attribut-, oder Attribut-zu-Attribut Co-occurrence Constraint.

Um die Stärken von RELAX NG zu demonstrieren, wird die zu modellierende Grammatik um eine solche wechselseitige Abhängigkeit erweitert: Das Inhaltsmodell des Elements `section` soll sich je nach Wert des `type`-Attributs unterschiedlich verhalten. Ist der Wert *global*, sind weitere `section`-Elemente oder aber `para`-Elemente erlaubt.

Ist der Wert *sub*, dürfen nur `para`-Elemente als Kinder auftreten. Eine solche Unterscheidung ist in RELAX NG-Grammatiken durchaus möglich, wie das Listing 3.52 zeigt (im Unterschied zu Listing 3.51 wurde auf das Laden der XSD-Datentypen-Bibliothek und das optionale `number`-Attribut verzichtet).

Listing 3.52: Erweiterte RELAX NG Grammatik

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
 datatypeLibrary="http://relaxng.org/ns/compatibility/datatypes/1.0">
 <start>
  <element name="text">
   <optional>
    <attribute name="author"/>
   </optional>
   <optional>
    <ref name="element.title"/>
   </optional>
   <oneOrMore>
    <choice>
     <ref name="element.section"/>
     <ref name="element.para"/>
    </choice>
   </oneOrMore>
  </element>
 </start>
 <define name="element.section">
  <choice>
   <oneOrMore>
    <element name="section">
     <optional>
      <ref name="element.title"/>
     </optional>
     <optional>
      <attribute name="type">
       <value>sub</value>
      </attribute>
     </optional>
     <oneOrMore>
      <ref name="element.para"/>
     </oneOrMore>
    </element>
   </oneOrMore>
   <oneOrMore>
    <element name="section">
     <optional>
      <ref name="element.title"/>
     </optional>
     <optional>
      <attribute name="type">
       <value>global</value>
      </attribute>
     </optional>
     <oneOrMore>
      <choice>
       <ref name="element.section"/>
       <ref name="element.para"/>
      </choice>
     </oneOrMore>
    </element>
   </oneOrMore>
  </choice>
 </define>
```

```
57  <define name="element.title">
58   <element name="title">
59    <text/>
60   </element>
61  </define>
62  <define name="element.para">
63   <element name="para">
64    <interleave>
65     <optional>
66      <attribute name="id">
67       <data type="ID"/>
68      </attribute>
69     </optional>
70     <optional>
71      <element name="xref">
72       <attribute name="href">
73        <data type="IDREF"/>
74       </attribute>
75      </element>
76     </optional>
77     <text/>
78    </interleave>
79   </element>
80  </define>
81 </grammar>
```

Der interessante Teil ist die Definition des `element.section`-Patterns in Zeile 20. Ein Pattern ist in etwa äquivalent zu einem benannten Inhaltsmodell (vergleichbar mit einer Parameter-Entität in XML-DTD bzw. einer benannten Model Group in XSD). Das in Zeile 20 deklarierte Pattern erlaubt innerhalb seines Inhaltsmodells das Auftreten zweier unterschiedlicher `section`-Elemente, die jeweils unterschiedliche Inhaltsmodelle haben – je nach Wert des optionalen `type`-Attributs.

Mit einer Umschreibung der entsprechenden Pattern-Definition lässt sich das noch deutlicher herausarbeiten: In Listing 3.53 gibt es nur noch ein `element`-Pattern, und die Auswahlmöglichkeit zwischen den beiden Unterinhaltsmodellen über den Wert des `type`-Attributs wurde weiter nach innen gezogen.

Listing 3.53: Umschreibung des `element.section`-Patterns

```
1  <!-- [...] -->
2  <define name="element.section">
3   <oneOrMore>
4    <element name="section">
5     <optional>
6      <ref name="element.title"/>
7     </optional>
8     <choice>
9      <group>
10      <optional>
11       <attribute name="type">
12        <value>sub</value>
13       </attribute>
14      </optional>
15      <oneOrMore>
16       <ref name="element.para"/>
17      </oneOrMore>
18     </group>
19     <group>
20      <optional>
21       <attribute name="type">
```

```
      <value>global</value>
     </attribute>
    </optional>
    <oneOrMore>
     <choice>
      <ref name="element.section"/>
      <ref name="element.para"/>
     </choice>
    </oneOrMore>
   </group>
  </choice>
 </element>
</oneOrMore>
</define>
<!-- [...] -->
```

Listing 3.54 zeigt zum Vergleich das vollständige Schema in der Compact Syntax.

Listing 3.54: Erweiterte RELAX NG Grammatik (Compact Syntax)

```
datatypes d = "http://relaxng.org/ns/compatibility/datatypes/1.0"

start =
  element text {
    attribute author { text }?,
    element.title?,
    (element.section | element.para)+
  }
element.section =
  element section {
    element.title?,
    ((attribute type { "sub" }?,
      element.para+)
     | (attribute type { "global" }?,
        (element.section | element.para)+))
  }+
element.title = element title { text }
element.para =
  element para {
    attribute id { d:ID }?
    & element xref {
        attribute href { d:IDREF }
      }?
    & text
  }
```

In XML SCHEMA verbietet die bereits in Abschnitt 3.4.3 erläuterte *Unique Particle Attribution* (UPA) in Kombination mit der *Consistent Declaration Rule* (in der deutschen Übersetzung als „Regel von der konsistenten Deklaration“ bezeichnet, van der Vlist 2003c, S. 122) die Definition eines solchen Inhaltsmodells.

Die Instanz aus Listing 3.55 ist valide gegenüber dieser erweiterten Dokumentgrammatik, während die in Listing 3.56 auf der nächsten Seite es nicht ist (ohne optionales `type`-Attribut wäre die Instanz aus Listing 3.55 weiterhin valide).

Listing 3.55: Valide XML-Instanz gegenüber der erweiterten RELAX NG-Grammatik

```
<?xml version="1.0" encoding="UTF-8"?>
<text author="ms">
 <title>A simple title</title>
 <section type="global">
  <title>A section title</title>
  <para id="p1">Introductory para</para>
```

```
  <section type="sub">
   <title>A subsection title</title>
   <para>Some text with a reference: <xref href="p1"/>.</para>
  </section>
 </section>
</text>
```

Listing 3.56: Ungültige XML-Instanz gegenüber der erweiterten RELAX NG-Grammatik

```
<?xml version="1.0" encoding="UTF-8"?>
<text author="ms">
 <title>A simple title</title>
 <section type="sub">
  <title>A section title</title>
  <para id="p1">Introductory para</para>
  <section type="sub">
   <title>A subsection title</title>
   <para>Some text with a reference: <xref href="p1"/>.</para>
  </section>
 </section>
</text>
```

Für die Diskussion von Algorithmen, die in der Lage sind, solche Co-occurrence Constraints zu verarbeiten und entsprechende Instanzen zu validieren, wird an dieser Stelle auf Hosoya und Murata 2002; Neven 2002; Murata und Hosoya 2003; Bouchou *et al.* 2003 verwiesen.

Die in Listing 3.52 auf Seite 97 gezeigte Dokumentgrammatik geht eindeutig über die Möglichkeiten von LTG und STG hinaus – es bleibt aber im weiteren Verlauf dieser Arbeit zu klären, ob die Nutzung von Co-occurrence Constraints nur mit der vollen Ausdrucksstärke einer *Regular Tree Grammar* möglich ist. Murata, D. Lee, Mani und Kawaguchi (2005) klassifizieren RNG-Grammatiken als RTG, was insgesamt zu einer Taxonomie der drei am häufigsten genutzten Grammatikformalismen führt. Wie in Stührenberg und Wurm (2010) und im Folgenden ausgeführt, kann vor allem die Klassifizierung von in RELAX NG formalisierten Grammatiken noch verbessert werden, da ein Großteil der in RELAX NG erstellten Schemata bei Weitem nicht das volle Potential nutzt. Doch zunächst soll die Definition einer weiteren Grammatikklasse aus Murata, D. Lee, Mani und Kawaguchi (2005) rezitiert werden, die der *Restrained-Competition Tree Grammar*.

3.4.4.5 Restrained-Competition Tree Grammar

Definition 6 *Eine* Restrained-Competition Tree Grammar (RCG) *ist eine reguläre Baumgrammatik, in der konkurrierende Regeln nicht im gleichen Inhaltsmodell und zusätzlich mit dem gleichen Präfix von Nichterminalen auftreten dürfen; daher sind Produktionsregeln mit identischer linker Seite, Terminalen und Inhaltsmodellen der folgenden Form nicht erlaubt:* $(\Gamma A \Delta)$ *und* $(\Gamma B \Delta')$*, wobei A und B konkurrierende Nichtterminale sind und griechische Großbuchstaben (auch leere) Sequenzen von Nichtterminalen referenzieren.*

Zu beachten ist, dass diese Einschränkung sich nur auf den linken Kontext konkurrierender Nichtterminale bezieht. Analog dazu kann auch eine Beschränkung des rechten Kontext definiert werden – allerdings sind beide von Murata, D. Lee, Mani und Kawaguchi aufgestellten Definitionen zu spezifisch, da sie beide unterschiedliche Klassen von

Sprachen erzeugen, die keine Inklusionsbeziehung aufweisen (weder in die eine noch in die andere Richtung), also keine Teilmengen voneinander sind. Dagegen weisen die vier skizzierten Grammatikklassen eine ordentliche Inklusionshierarchie auf: LTGs sind STGs, STGs sind RCGs, und diese sind RTGs (umgekehrt gilt diese Teilmengenrelation natürlich nicht).

Des Weiteren wird (ebenfalls Murata, D. Lee, Mani und Kawaguchi 2005, S. 663 folgend) die Interpretation eines Baumes gegen eine dazu gehörige Grammatik wie folgt definiert:

Definition 7 *Eine Interpretation I eines Baumes t gegen eine Grammatik G ist eine Abbildung jedes Knotenlabels von t, dargestellt durch e, auf ein Nichtterminal N der Grammatik, so dass gilt:*

- *I(e) ist ein Startsymbol, wenn e die Wurzel des Baumes t ist,*
- *für jedes e und dessen Töchterknoten $e_0, e_1, ..., e_n$ gibt es in G eine Produktionsregel $A \rightarrow a(r)$, so dass gilt:*
  - *I(e) ist A,*
  - *das Label von e ist a,*
  - *$I(e_0), I(e_1), ..., I(e_n)$ ist eine Instanziierung von r.*

Ein Baum t ist valide gegenüber einer Grammatik G, sofern es eine Interpretation von t gegen G gibt. Wichtig ist in diesem Zusammenhang erneut der Hinweis, dass zwischen Knotenlabel und Interpretation eines Knotens zu unterscheiden ist. Das Label im Baum entspricht dem Terminalsymbol in der Produktionsregel der Grammatik und ist im Baum direkt sichtbar, während die Interpretation eines Knotens das Nichtterminal bezeichnet, durch das das Knotenlabel erzeugt wurde. Dieses Nichtterminal ist nicht im Baum sichtbar und kann nur durch Inferenz erschlossen werden. Zusätzlich muss zwischen der Regel als solcher und ihrer Instanziierung unterschieden werden (analog zu einem Schema und seiner XML-Instanz): Da Inhaltsmodelle aus regulären Ausdrücken über Nichtterminale bestehen, beziehen sie sich auf Mengen von Sequenzen von Nichtterminalen, wobei jede Sequenz (jede Instanziierung) ein Element dieser Menge ist, die der reguläre Ausdruck denotiert. Ebenfalls problematisch ist die Tatsache, dass es nicht notwendigerweise eine 1:1-Beziehung zwischen Nichtterminalen und Terminalen (also Labeln) gibt; als Konsequenz kann eine Abfolge von Labels durch verschiedene Instanziierungen eines Inhaltsmodells generiert werden, was für Ambiguität sorgt.

Die Ausführungen bis zu diesem Punkt haben die bereits bekannten Konzepte rekapituliert, die im Folgenden weiterentwickelt werden sollen.

3.4.4.6 Das Verhältnis von Determinismus und lokaler Ambiguität

Determinismus (oder das Fehlen davon) ist nicht nur abhängig vom verwendeten Grammatikformalismus, sondern auch vom verwendeten Algorithmus. In Regulären Baumsprachen (*Regular Tree Languages, RTL*) gibt es kein eindimensionales Konzept von Determinismus, wie es für Sprachen regulärer Zeichenketten (*Regular String Languages,*

RSL) existiert, da diese als eindimensionale Strukturen aufgefasst werden können (was die Verarbeitung erleichtert). Für die adäquate Analyse von Baumstrukturen oder den Zeichenketten, die von deren Bäumen formuliert werden, werden mindestens zwei Dimensionen benötigt (vgl. Rogers 2003).

Wie bereits ausgeführt, müssen Inhaltsmodelle sowohl in DTD als auch in XML SCHEMA deterministisch (im Sinne von Brüggemann-Klein und Wood (1997a)) sein, nur RELAX NG macht hier eine Ausnahme. Die Gründe dafür liegen in der Verarbeitbarkeit: Sofern eine Grammatik deterministisch ist, kann das Parsen deutlich effizienter erfolgen, da kein Vorhalten von Informationen (z. B. Backtracking oder Speicherung nicht-lokaler Informationen) notwendig ist. Allerdings haben bereits Clark (2002) und Murata, D. Lee, Mani und Kawaguchi (2005) Algorithmen vorgestellt, die (z. B. mittels Baum-Automaten) das effiziente Parsen von RTGs erlauben, auch Hosoya (2010) diskutiert eine Reihe von Algorithmen zur Verarbeitung von Baumstrukturen inkl. Typechecking.

Generell ist das Konzept von Determinismus eng verbunden mit dem Konzept lokaler Ambiguität: Sofern keine lokale Ambiguität vorliegt, ist in jedem Punkt des Parsingprozesses die Struktur bekannt, d. h., der jeweils nächste Folgeknoten sowie der direkt vorhergehende im Baum (rechte bzw. linke Geschwister) sind eindeutig bestimmbar. Liegt lokale Ambiguität vor, entsteht Nicht-Determinismus: Es kann nicht mehr eine eindeutige Interpretation getroffen werden, da dazu Informationen benötigt werden, die (noch) nicht verfügbar sind.

3.4.4.7 Deterministische Inhaltsmodelle

Das Konzept des deterministischen Inhaltsmodells, aufbauend auf der Betrachtung von deterministischen (d. h. 1-nicht-ambigen) regulären Ausdrücken (*Deterministic Regular Expression, DRE*) durch Brüggemann-Klein und Wood (1997a) kann wie folgt definiert werden:[55]

Definition 8 *Sei s eine Zeichenkette, die vom regulären Ausdruck E denotiert wird, wobei unsere regulären Ausdrücke mit Buchstaben des Alphabets Π erstellt werden und Π identisch zum Alphabet Σ ist, mit dem Unterschied, dass zusätzliche Indizes für die Buchstaben (bestehend aus natürlichen Zahlen) zur Verfügung stehen. Beim Erstellen eines Ausdrucks aus Π ist jeder Index maximal einmal vertreten (vergleichbar mit der UPA in XSD). Davon abgesehen gelten die gewöhnlichen Konstruktoren für reguläre Ausdrücke. Die Indizes bleiben erhalten d. h., sie sind nicht nur im regulären Ausdruck präsent, sondern auch in jedem Buchstaben der erzeugten Zeichenkette. Ein Buchstabe a_i aus s instanziiert einen Buchstaben a_j aus E, genau dann, wenn $i = j$.*

Der Index ermöglicht für jede durch einen regulären Ausdruck erzeugte Zeichenkette eine eindeutige Abbildung von Buchstaben aus s auf Buchstaben aus E. Definiert wird nun eine Abbildung $\natural$ für Zeichenketten aus Π zu solchen aus Σ, so dass gilt: $\natural(x_i) := x$, $\natural(xs) := \natural(x)\natural(s)$, wobei s eine Zeichenkette ist und x ein Buchstabe (genauer: der erste Buchstabe der Zeichenkette). Dieser Homomorphismus erlaubt das Löschen der

[55] Vgl. dazu auch Mani (2001) für eine Diskussion der Vor- und Nachteile 1-nicht-ambiger Inhaltsmodelle in XML-Schemasprachen.

Indizes ohne sonstige Auswirkungen. Zu beachten ist, dass für eine Zeichenkette $\natural s$ und einen Ausdruck E es mehrere $s \in E$ geben kann, sofern die Abbildung unter $\natural$ identisch ist. In diesem Fall spricht man davon, dass ein Zeichen in s mehrere Zeichen in E instanziieren kann. Zur Erinnerung sei an dieser Stelle die Definition eines deterministischen regulären Ausdrucks über Π wiederholt (Brüggemann-Klein und Wood 1997a, S. 185 und Abschnitt 3.4.3):

Definition 9 *E ist deterministisch (oder 1-nicht-ambig), wenn für alle Zeichenketten u, v, w über Π, und alle Buchstaben x, y in Π gilt: Wenn uxv und $uyw \in L(E)$, und $x \neq y$, dann gilt auch $\natural x \neq \natural y$.*

Das bedeutet, dass die Indizes unberücksichtigt bleiben können und man dennoch nur durch Kenntnis des linken Kontexts feststellen kann, welches Zeichen in E durch welches Zeichen in s instanziiert wurde. Formal gesehen bedeutet das, dass die Abbildung $\natural$ reversibel ist. Eine reguläre Sprache ist deterministisch, wenn sie durch einen deterministischen regulären Ausdruck bezeichnet werden kann. Ein einfaches Beispiel eines nicht-deterministischen Ausdrucks ist der Ausdruck $a^*b^*a^*$, da für das Zeichen a nicht entscheidbar ist, ob es die Instanziierung des ersten oder zweiten as ist (für andere Beispiele sei auch hier wieder auf den Abschnitt 3.4.3 verwiesen).

In der Automatentheorie korrespondieren deterministische reguläre Ausdrücke mit deterministischen Glushkov-Automaten (vgl. Glushkov 1961; Caron und Ziadi 2000), wobei durch die Restriktion auf 1-nicht-ambige Inhaltsmodelle sichergestellt wird, dass der entsprechende Glushkov-Automat deterministisch ist. Allerdings verbleibt ein Problem, wenn dieses Konzept auf die Inhaltsmodelle regulärer Baumgrammatiken übertragen wird: Man kann nicht von einem Knotenlabel im Baum auf ein eindeutiges Nichtterminal zurück schließen: Sofern konkurrierende Regeln vorliegen, können unterschiedlichen Nichtterminale identische Knotenlabel einführen.

Als Beispiel soll die folgende Grammatikregel dienen:

$$A \rightarrow a(ABC|CBA)$$

Das Inhaltsmodel ist deterministisch. Aber falls A und C konkurrierende Regeln sind (also identische Labels haben), wird die Festlegung einer eindeutigen Interpretation dennoch erschwert. Jeder Teilbaum muss überprüft werden bis eine eindeutige Interpretation gefunden ist. Das kann bedeuten, dass beide Teilbäume vollständig überprüft werden müssen (z. B. in dem Fall, dass beide Bäume, die durch $A(C)$ erzeugt werden eine Untermenge der Bäume ergeben, die durch $C(A)$ generiert werden können).

Die nahe liegende Ursache für die Beschränkung dieses Konzepts von Determinismus ist dessen Herkunft aus den eindimensionalen Zeichenketten. Da die Bäume, die im Laufe dieser Arbeit betrachtet werden sollen, aber zweidimensional sind, kann Determinismus nur unter Berücksichtigung der Parsingstrategie definiert werden; also entweder *Top-Down*- oder *Bottom-Up*-Verarbeitung.[56]

[56] An dieser Stelle wird auf den Unterschied zwischen *Depth-First* (Tiefensuche) und *Breadth-First* (Breitensuche) nicht weiter eingegangen.

Aufgrund der gemachten Ausführungen wird im Folgenden die Murata-Hierarchie im Hinblick auf die Art von Determinismus erweitert, die durch den jeweiligen Grammatikformalismus eingeführt wird. Dabei wird das eindimensionale Problem nichtdeterministischer Inhaltsmodelle nicht weiter bearbeitet, da dieses schon hinlänglich von Brüggemann-Klein und Wood (1997a) analysiert wurde.

3.4.4.8 Die Murata-Hierarchie als Hierachie lokaler Bedingungen

Da das Hauptaugenmerk auf den formalen Eigenschaften von XML-Grammatikformalismen (und deren Auswirkungen auf die Verarbeitung) liegt, wird zunächst die von Murata, D. Lee, Mani und Kawaguchi (2005) gemachte Taxonomie mit dem Ziel umformuliert, deutlich zu machen welche Informationen benötigt werden, um eine eindeutige Interpretation eines lokalen Knotens oder Teilbaums zu erhalten.

In einer lokalen Baumgrammatik (Local Tree Grammar, LTG) ist für jeden Knoten a in jedem Kontext eine eindeutige Interpretation bekannt. Diese Aussage ist offensichtlich, da es für jedes Knotenlabel nur eine einzige Regel gibt, die es generiert. Als Konsequenz ist das Parsen deterministisch und damit ist die Zuordnung von Knoten zu Interpretation in einer konstanten Abfolge von Schritten möglich.

In einer Single Typ Tree Grammar (STG) ist für jedes Knotenlabel A in jedem Kontext eine eindeutige Interpretation bekannt, sofern die Interpretation seines Mutterknotens bekannt ist (diese Aussage folgt direkt aus der Definition). Zu beachten ist allerdings, dass es nicht ausreicht, das Label des Mutterknotens zu kennen, was sich durch das folgende Beispiel zeigen lässt: Es gibt zwei konkurrierende Regeln $A \rightarrow a(C)$ und $B \rightarrow a(D)$, deren Nichtterminale nicht im gleichen Inhaltsmodell einer Regel auftreten. Darüber hinaus gibt es die beiden Regeln $C \rightarrow b(r)$ und $D \rightarrow b(r')$. Beide Knoten, eingeführt durch C und D, haben das Knotenlabel b und deren Mutterknoten haben beide das Label a, obwohl sie unterschiedliche Interpretationen haben.

Das hat direkte Auswirkungen auf die Verarbeitung: Ein Top-Down-Parser kennt an jedem Punkt die Interpretation eines Knotens, bei dem Bottom-Up-Verfahren kann es vorkommen, dass für eine korrekte Interpretation zunächst ein Aufstieg bis zum Wurzelknoten erfolgen muss. Das Finden der Interpretation eines gegebenen Knotens ist damit bei der Top-Down-Strategie ein lineares Suchproblem, da für einen gegebenen Baum und einen gegebenen Knoten es immer ausreicht, einen Pfad von der Wurzel bis zu diesem Knoten zu verfolgen.

In einer Restrained Competition Grammar (RCG) wird für die eindeutige Interpretation eines Knotens die Interpretation seines Mutterknotens und gegebenenfalls seine linken Geschwisterknoten (sofern vorhanden) benötigt. In dieser Definition von RCGs ist nur das Wissen über die Label der Geschwisterknoten notwendig, nicht deren Interpretation, da jegliche zwei konkurrierenden Nichtterminale innerhalb des gleichen Inhaltsmodells durch ein eindeutiges Präfix (im Sinne von linken Geschwisterknoten) unterscheidbar sind, das selbst weder aus konkurrierenden Nichtterminalen bestehen noch leer sein darf. Aufgrund dieser Einschränkung bleibt die Grammatik unambig und damit leichter zu verarbeiten. Allerdings gehen damit einige weitere Einschränkungen einher, die man evtl. nicht hinnehmen möchte, z. B. wenn die eindeutige Interpretation eines Labels nicht von dem vorhergehenden (linken), sondern von dem folgenden (rechten) Geschwister-

knoten abhängen soll. So können in RCGs konkurrierende Nichtterminale nicht als ganz linkes Symbol innerhalb des Inhaltsmodells auftreten, aber als ganz rechtes, sofern linker Kontext besteht. Dies erzeugt eine Asymmetrie, die künstlich erscheint. Natürlich ist es möglich, einen entsprechenden asymmetrischen Grammatiktyp zu definieren, der auf eindeutige Suffixe (rechte Geschwisterknoten) prüft. Allerdings löst das nicht das generelle Problem. Wünschenswert wäre eine generalisierte Grammatikklasse, die sowohl eindeutige Präfixe als auch Suffixe berücksichtigt, eine *Generalized Restrained Competition Grammar*.

Eine solche Klasse kann nun wie folgt definiert werden:

Definition 10 *In einer* Generalized Restrained Competition Grammar (GRCG) *schlägt für zwei konkurrierende Nichtterminale A und B innerhalb des selben Inhaltsmodells entweder* $(\Gamma A \Delta)$ *oder* $(\Gamma B \Delta)$ *als Entsprechung zu r fehl (griechische Variablen erstrecken sich über potentiell leere Sequenzen von Nichtterminalen).*

Mit dieser Definition wurde die Einschränkung vom linken (bzw. rechten) Rand (Präfix, resp. Suffix) generalisiert auf den gesamten Kontext. Zu beachten ist, dass dadurch auch die Restriktion bzgl. des Inhaltsmodells genauer spezifiziert wird – im Effekt besteht zwischen GRCGs und RCGs eine ordentliche Teilmengenbeziehung.
Nicht verschwiegen werden soll allerdings, dass diese Generalisierung nur unter erheblicher Steigerung der Verarbeitungskomplexität möglich ist, da für diesen Grammatiktyp zum Erreichen einer eindeutigen Interpretation eines gegebenen Knotens a in seinem Kontext im schlimmsten Fall die Interpretation des Mutterknotens, seiner Geschwisterknoten und die seiner Kind-Teilbäume notwendig sind. Und selbst dann können GRCGs weiterhin ambig sein, d. h. mehr als eine Interpretation für einen Baum ermöglichen.

Zur Erklärung: Dass die Interpretation des Mutterknotens benötigt wird, lässt sich folgern aus dem Argument, dass RCGs (und damit auch GRCGs) Oberklassen von STGs darstellen. Aber nur diese ist nicht ausreichend, da konkurrierende Nichtterminale auch innerhalb des gleichen Inhaltsmodells auftreten dürfen (vgl. Listing 3.52 auf Seite 97). Daher müssen alle Geschwisterknotenlabel auf die Nichtterminale des Inhaltsmodells der Interpretation des Mutterknotens abgebildet werden, um eine eindeutige Interpretation zu erhalten. Allerdings werden nicht nur die Label der Geschwisterknoten sondern deren Interpretation benötigt. Dies kann durch die folgende Grammatikregel verifiziert werden: $A \rightarrow a(BC|B'C')$, wobei B und B' und C und C' konkurrierende Regeln sind, die die Label b und c einführen. Damit wären alle Bedingungen der GRCGs erfüllt. Für die korrekte Interpretation der Label b und c reichen nicht die Label der Geschwisterknoten aus, es werden deren Interpretationen benötigt.
Die Anforderungen an eine eindeutige Interpretation lassen sich weiter steigern, wenn die Interpretation eines Geschwisterknotens abhängig ist von der Interpretation des Knotens unter Betrachtung, wie im Beispiel $A \rightarrow a(BC|CB)$, wobei B und C konkurrierende Regeln sind.
Offensichtlich sind beide abhängig von ihrem Geschwisterknoten innerhalb des Inhaltsmodells von A (C kann nur auftreten mit B auf der linken Seite oder auf der rechten Seite). Da sie aber die gleichen Knotenlabel haben, reicht es nicht aus, die Label des Geschwisterknotens zu überprüfen (da sie identisch sind). Des Weiteren kann der

Fall konstruiert werden, dass es unmöglich ist, eine Interpretation eines Teilbaums zu erhalten ohne die der Geschwister (z. B. wenn eines der beiden konkurrierenden Nichtterminale eine Teilmenge der Bäume des anderen Nichtterminals erzeugt).

Aus Sicht der Verarbeitungskomplexität ist festzuhalten, dass die Ausdrucksstärke von GRCGs nur zu einem hohen Preis zu haben ist: Weder ein Bottom-Up- noch ein Top-Down-Parser können lokal eine eindeutige Interpretation bestimmen, evtl. sogar nicht global. Die Zuordnung eines gegebenen Knotens zu einer eindeutigen Interpretation kann daher als ein exponentielles Suchproblem angesehen werden – evtl. ist sie gar nicht entscheidbar.[57] Deshalb erscheint es ratsam, einen Subtyp von GRCGs einzuführen, genannt *Unambigious Restrained Competition Grammar* (*URCG*). Dieser Typ hat ebenfalls eine ordentliche Inklusionsbeziehung zur Klasse der GRCG (ist also eine echte Teilmenge) und ist seinerseits Superklasse der STGs und RCGs. Was erreicht werden soll, ist der Ausschluss der Ambiguität.

In einer URCG soll es möglich sein, eine eindeutige Interpretation eines Knotens durch die Interpretation seines Mutterknotens und der Label (nicht Interpretationen) seiner Geschwisterknoten zu erreichen. Dieser Grammatiktyp lässt sich mittels folgender Terme charakterisieren: Eingeführt wird ein Alphabet von Meta-Variablen O ein, das wie folgt genutzt wird: Gebildet wird eine Menge von Mengen von Nichtterminalen aus N, die miteinander konkurrieren; diese (möglicherweise auch nur einelementige) Mengen werden *Competition Sets* genannt. Dabei sind besonders die Mengen interessant, in denen jedes Nichtterminal in genau einem Competition Set auftritt, so dass die Menge eine Partition von N ergibt. Über die Gesamtmenge der möglichen Partitionen kann mit folgender Prozedur iteriert werden: Jedem Competition Set wird ein Symbol aus O zugewiesen (*O-Assignment*). Anschließend wird für alle Nichtterminale in allen Inhaltsmodellen kontrolliert, ob das Inhaltsmodell weiterhin die GRCG-Bedingungen erfüllt, wenn alle anderen Nichtterminale durch die entsprechenden Symbole aus O ersetzt werden. Sofern es mehr als eine mögliche Zuweisung gibt, d. h., wenn ein Nichtterminal in mehr als einem Competition Set auftritt, muss diese Prozedur über alle möglichen Zuweisungen iterieren. Wenn für alle Zuweisungen, sowohl Nichtterminale als auch Inhaltsmodelle, die resultierende Grammatik eine GRCG ist, dann ist die ursprüngliche Grammatik eine URCG. Zu beachten ist, dass diese Zuweisungsprozedur nur für den Zweck der Evaluation einer Grammatik eingeführt wird.

Kontexte, die auch unter *O-Assignment* identisch sind, werden **ähnlich** genannt. Dementsprechend ist eine URCG als eine Grammatik definiert, in der konkurrierende Nichtterminale nicht im gleichen Inhaltsmodell und nicht in ähnlichen (und damit auch nicht in identischen) Kontexten auftreten dürfen. Damit ist sichergestellt, dass konkurrierende Nichtterminale nicht im gleichen Kontext der Knotenlabel auftreten dürfen (im Gegensatz zu Nichtterminalen). Damit ist eine URCG eine GRCG, in der für jeden Knoten eine eindeutige Interpretation gefunden werden kann, sofern die Interpretation des Mutterknotens und die Label der Geschwisterknoten bekannt sind. Das Inhaltsmodell $(BC|CB)$, in dem B und C konkurrierende Nichtterminale sind,

[57] Ein solches Vorliegen muss nicht zwangsläufig komputationell als negativ angesehen werden – vorstellbar ist auch ein Szenario, in dem sehr schnell festgestellt werden kann, dass eine Zuordnung nicht entscheidbar ist, was zu einem schnellen Abbruch und eben nicht zu exponentiellen Laufzeiten führt.

erfüllt nicht diese Bedingung, da sowohl *B* als auch *C* im gleichen Kontext __*X* bzw. *X*__ auftreten, wobei *X* das entsprechende *O-Assignment* für beide Nichtterminale ist.

Dieser Grammatiktyp ist aus einem Grund für die Betrachtung von XML-Schemasprachen sinnvoll: Er beinhaltet keine globale Ambiguität (da die einzige Quelle für Ambiguitäten, die gegenseitige Abhängigkeit der Interpretation von Knotenlabeln, entfernt wurde); und er ist darüber hinaus der ausdrucksstärkste d. h. mächtigste Typ der nicht-ambigen Grammatiktypen, die bisher betrachtet wurden. Für das Finden einer eindeutigen Interpretation eines Knotens kann es zwar weiterhin notwendig sein, alle Geschwisterknoten zu überprüfen, es reichen aber deren Knotenlabels aus. URCGs stellen eine Superklasse der RCGs dar, und auch das Suchproblem für URCGs bleibt weiterhin linear, da nur der Pfad von der Wurzel bis hin zum gegebenen Knoten und zusätzlich eine endliche Menge an Knotenlabels der Geschwisterknoten überprüft werden müssen.

Ein Beispiel für die Verwendung von URCGs in XML-Schemasprachen ist das in Listing 3.52 auf Seite 97 gezeigte Attribute-Element Co-occurrence Constraint zwischen dem Wert des `type`-Attributs und dem Inhaltsmodell des `section`-Elements. Weitere Beispiele finden sich in van der Vlist (2003b, Kapitel 7.2, „Co-Occurrence Constraints“), Clark, Cowan *et al.* (2003, Abschnitt 15, „Non-restrictions“) oder auch im RELAX-NG-Schema der *Japanese Layout Taskforce* (JLTF) für die Deklaration der `div`-Elemente der Klasse „Taiyaku“ – in der Dokumentgrammatik durch die Pattern `div1heading`, `div2heading` und `div3heading` repräsentiert (vgl. Sasaki 2010). Auch die in Teil III diskutierte Meta-Auszeichnungssprache XStandoff nutzt Co-Constraints, hier für die erweiterte Überprüfung der Instanzen.

Weder DTD noch XML Schema in der aktuellen Produktivversion 1.0 unterstützen die gezeigten Attribut-Element-Constraints, allerdings ist es in XSD 1.0 möglich, ein analoges Verhalten zu simulieren; darunter fallen die Verwendung von `xsi:type`-Attributen oder von `xs:key`-Elementen (vgl. van der Vlist 2003b, S. 65; van der Vlist 2007, S. 45ff.). Die bereits in Abschnitt 3.4.4.2 erwähnte Spezifikation DTD++ 2.0 unterstützt eine große Zahl von Co-Constraints, hat aber den Status einer Forschungsinitiative, die nicht weiter entwickelt wird. Eine andere Möglichkeit, eine Wechselwirkung zwischen Attributwert und Inhaltsmodell zu modellieren, ist die Nutzung eingebetteter Schematron *Business Rules* (vgl. Robertson 2002) oder das mit XML Schema 1.1 eingeführte *Conditional Type Assignment* – entweder durch Verwendung von *Type Alternatives* oder aber *Assertions*, wobei letztere den Schematron Business Rules nachempfunden sind.[58] Sowohl Schematron Business Rules (in der aktuellen Version) als auch XSD 1.1 Assertions kommen in XStandoff zum Einsatz (letztere in der Entwicklerversion, vgl. Kapitel 13.4.1), um die dortigen Co-Constraints umzusetzen. Listing 3.57 zeigt eine Beispiel-Realisierung mit Hilfe des in XSD 1.1 enthaltenen `assert`-Elements (Zeile 27). Über den XPath-Ausdruck als Wert des `test`-Attributs werden die möglichen Kindelemente abhängig vom Wert des `type`-Attributs restringiert.

Listing 3.57: Grammatik auf Basis von XML Schema 1.1

```
1 <?xml version="1.0" encoding="UTF-8"?>
```

[58] Für detailliertere Ausführungen vgl. Peterson *et al.* (2012) für XML Schema *simpleTypes* und Gao *et al.* (2012) für XML Schema *complexTypes*.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="text">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="textType">
     <xs:attribute name="author" type="xs:string" use="optional"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="title" type="xs:string"/>
 <xs:element name="section">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="title" minOccurs="0"/>
    <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="type" use="optional">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="global"/>
      <xs:enumeration value="sub"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:assert test="@type!='sub' and (child::para | child::section) or @type='sub' and not(
       child::section)" />
  </xs:complexType>
 </xs:element>
 <xs:element name="para">
  <xs:complexType mixed="true">
   <xs:sequence>
    <xs:element name="xref" minOccurs="0">
     <xs:complexType>
      <xs:attribute name="href" type="xs:IDREF" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute ref="id" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:complexType name="textType">
  <xs:sequence>
   <xs:element ref="title" minOccurs="0"/>
   <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:group name="sectOrPara">
  <xs:choice>
   <xs:element ref="section"/>
   <xs:element ref="para"/>
  </xs:choice>
 </xs:group>
 <xs:attribute name="id" type="xs:ID"/>
</xs:schema>
```

Ein Unterschied zwischen den in Listing 3.52 auf Seite 97 mittels RELAX NG dargestellten Co-occurrence Constraints und der hier gezeigten Verwendung des `assert`-Elements besteht darin, dass derlei Restriktionen in XSD 1.1 weiterhin auf direkte Kindknoten beschränkt sind (d. h. das `assert`-Element nutzt einen Teilbaum, dessen

Wurzelelement das Element ist, gegen das es getestet wird).[59] Darüber hinaus existiert die Einschränkung, dass die volle XPATH-Unterstützung (also inkl. Achsen wie *ancestor*, *parent*, sowie *preceding/preceding-sibling* respektive *following/following-sibling*) abhängig von der gewählten Implementierung ist. Es ist aber möglich, das `assert`-Element auf der Ebene eines gemeinsamen Elternknotens zu platzieren, also der Hierarchieebene des Elements, das alle relevanten Informationen (und Kindknoten) beinhaltet. Damit wächst naturgemäß die Anzahl der Knoten, die verarbeitet werden müssen, um die Testbedingung zu prüfen.

Auch XML SCHEMA 1.1 Type Alternatives beschränken sich auf Tests gegen Konstanten oder Attribute des gleichen Elements, nicht aber auf Vorfahren, Nachkommen oder Geschwister. Diese Einschränkung in XSD 1.1 ist durchaus beabsichtigt: Auch die aktuelle Version der Spezifikation zielt darauf ab, die Validierung von Typen kontextfrei zu gestalten, d. h., der Typ eines Elements sollte sich auch bestimmen lassen, wenn es nicht mehr Teil einer XML-Instanz ist – eine Eigenschaft, die auch die Verarbeitung durch XPATH, XSLT und XQUERY stark erleichtert.[60] Insgesamt bleiben die Möglichkeiten von XML SCHEMA 1.1 damit weiterhin hinter der Ausdrucksmächtigkeit von SCHEMATRON Business Rules zurück. Allerdings war von Seiten der Entwickler nie beabsichtigt SCHEMATRON zu ersetzen. Es bleibt als Vorteil der kommenden Version von XML SCHEMA die Möglichkeit, innerhalb einer Technologie Vorgaben bzgl. der Struktur und der Datentypen mit zusätzlichen Einschränkungen kombinieren zu können – was so z. Zt. keine andere Constraint Language bietet.

Die gezeigten Co-occurrence Constraints erweitern nicht die formalen generativen Kapazitäten von URCGs. Das liegt darin begründet, dass Attribute bzw. deren Werte zusätzliche Informationen zu den *Terminalen* hinzufügen. Dabei lassen sich konkurrierende Terminale (äquivalent: Regeln) in nicht-konkurrierende umwandeln – nicht aber umgekehrt. Jeder Co-occurrence Constraint kann daher dazu führen, Nichtterminale unterscheidbar zu machen, die ansonsten nicht unterscheidbar wären; dadurch bliebt die Komplexität der zu verarbeitenden Grammatik entweder konstant oder sinkt sogar noch.

3.4.5 Anwendung und Verbreitung

In diesem Abschnitt soll kurz auf die derzeitige Verbreitung und Software-Unterstützung der diskutierten Grammatikformalismen eingegangen werden. Die bereits in van der Vlist (2001) zu findenden Ergebnisse haben auch weiterhin Bestand: Die Constraint Language mit der stärksten Verbreitung ist unbestritten die in der XML Spezifikation enthaltene DTD. DTDs lassen sich – auch aufgrund ihrer geringen formalen Ausdrucksstärke – nicht nur einfach entwickeln, auch entsprechende Parser sind schnell implementiert. Jegliche XML-unterstützende Software (seien es Editoren oder validierende Parser oder XSLT-Prozessoren) kann mit Dokumentgrammatiken, die mit diesem Formalismus erstellt wurden, umgehen. XML SCHEMA und RELAX NG werden ebenfalls von einer großen

[59] SCHEMATRON Business Rules sind prinzipiell nicht beschränkt auf ein XPATH-Subset.

[60] Es ist natürlich zu beachten, dass aufgrund der oben aufgeführten Assertions in XSD 1.1 die Überprüfung eines Typs nicht mehr ganz kontextfrei sein kann, sofern ein diesen Typ einschränkendes `assert`-Element bei einem gemeinsamen Vorfahren im Baum positioniert wird.

Anzahl an Software-Produkten unterstützt, wobei ein momentan evtl. noch vorhandener Vorsprung von XSD mehr und mehr ausgeglichen wird (belastbare Zahlen liegen hierzu nicht vor). Die folgende Liste erhebt keinerlei Anspruch auf Vollständigkeit. Die Editoren EMACS[61] (mit dem nXML-Modus[62]) und XMLOPERATOR[63] unterstützen die Validierung einer Instanz gegen ein RELAX-NG-Schema. Andere Produkte, wie LIQUID XML STUDIO[64], STYLUS STUDIO[65] oder ALTOVA XML SPY[66] dagegen können Instanzen nur gegen XSDs (und natürlich DTDs) überprüfen; <OXYGEN/>[67] unterstützt sowohl RELAX NG als auch XSD. Darüber hinaus stehen verschiedene validierende Parser zur Verfügung, die Instanzen gegen beide Constraint Languages prüfen können, allen voran der ORACLE MULTI-SCHEMA XML VALIDATOR (MSV)[68].

Auffallend ist, dass mehr und mehr Annotationsformate RELAX NG als primären Grammatikformalismus wählen (wobei oftmals noch zusätzliche Grammatikformalismen unterstützt werden), so z. B. DOCBOOK (vgl. Abschnitt 4.3) oder auch die TEI (vgl. Kapitel 5). Zu beachten ist dabei, dass diese Spezifikationen aufgrund der gleichzeitigen Unterstützung mehrerer Grammatikformalismen nicht die vollständige formale Ausdrucksstärke von RELAX NG nutzen können, sondern entweder auf STG- oder sogar LTG-Niveau zurückbleiben.

Die Unterstützung von Co-Occurrence Constraints ist abhängig vom gewählten Grammatikformalismus: Eingebettete SCHEMATRON-Regeln werden nur selten korrekt interpretiert, einfacher ist oft die Nutzung mittels XSLT-Prozessoren, da sich SCHEMATRON-Regeln in XSLT-Stylesheets umwandeln lassen. Die in XSD 1.1 enthaltenen `assert`-Elemente werden aktuell vom Schema-Parser des XSLT-Prozessors SAXON ab der Version 9.2 in der kommerziellen Variante SAXON-EE[69] und XERXES-J[70] ab Version 2.10 unterstützt.

3.4.6 Beurteilung der Grammatikformalismen

Die drei am häufigsten vertretenen Grammatikformalismen, DTD, XML SCHEMA und RELAX NG lassen sich anhand ihrer Ausdrucksstärke auf die drei Klassen Local Tree Grammar, Single Type Tree Grammar und Regular Tree Grammar abbilden – wobei hier die bereits in Abschnitt 3.3.5 gemachte Annahme gilt, dass interne Verknüpfungsmechanismen zur Erstellung hierarchie-unabhängiger Kanten (Integritätsmerkmale, Zeile „Integrität“ in Tabelle 3.3 auf der nächsten Seite) außer acht bleiben (vgl. Møller und Schwartzbach 2006a, S. 184), sowie dass bei XML SCHEMA auf den Einsatz von Wildcards verzichtet wird (vgl. Abschnitt 3.4.4.3).

[61] Vgl. `http://directory.fsf.org/project/emacs/`, zuletzt abgerufen am 19.04.2012.
[62] Vgl. `http://www.thaiopensource.com/nxml-mode/`, zuletzt abgerufen am 19.04.2012.
[63] Vgl. `http://www.xmloperator.net/`, zuletzt abgerufen am 19.04.2012.
[64] Vgl. `http://www.liquid-technologies.com/xml-studio.aspx`, zuletzt abgerufen am 19.04.2012.
[65] Vgl. `http://www.stylusstudio.com/xml_product_index.html`, zuletzt abgerufen am 19.04.2012.
[66] Vgl. `http://www.altova.com/xmlspy.html`, zuletzt abgerufen am 19.04.2012.
[67] Vgl. `http://www.oxygenxml.com`, zuletzt abgerufen am 19.04.2012.
[68] Vgl. `http://msv.java.net/`, zuletzt abgerufen am 19.04.2012.
[69] Vgl. `http://www.saxonica.com`, zuletzt abgerufen am 19.04.2012.
[70] Vgl. `http://projects.apache.org/projects/xerces_for_java_xml_parser.html`, zuletzt abgerufen am 19.04.2012.

Tabelle 3.3: Beurteilung der Constraint Languages

	XML-DTD	XSD	RELAX NG	SCHEMATRON
Herausgeber	W3C		OASIS/ISO	ISO
CL-Typ	Grammatik[1]			Regelbasiert
Grammatikklasse	LTG	STG/RCG[2]	RTG/URCG	×
Status	W3C Recommendation[3]		ISO-Standard	
Syntax†	Eigene	XML	XML/Eigene	XML
PSVI	✓ (−−)	✓	×	×
Strukturen	✓	✓	✓	×[4]
Datentypen	✓ (−−)	✓	✓ (−−)[5]	×[6]
Integrität	✓[7]	✓[8]	×[9]	×[10]
Inhaltsmodell	Deterministisch[11]		Nicht-deterministisch[12]	
Inklusion	✓ (−−)	✓	✓	✓
XML Namespace	×	✓	✓	✓
Verbreitung	++	+	+	o
Software-Support	++	++	+ (steigend)	−[13]
Anmerkung	Eigene Syntax.	Offizieller W3C Schema-Standard.	Starke mathematische Fundierung	Sprache zur Definition kontextabhängiger Regeln. Einbettung in XSD und RELAX NG möglich.

✓/× – Merkmal vorhanden/nicht vorhanden.
Ausprägung: ++ – sehr stark/+ – stark/o – zufriedenstellend/− – schwach/−− – sehr schwach.
† „XML“ bedeutet die Serialisierung der Grammatik in der Notation von XML-Instanzen, „Eigene“ die Nutzung einer proprietären Notation.

[1] XSD kann sowohl als grammatikbasiert als auch als objektorientiert eingestuft werden (van der Vlist 2003c, S. 372); RELAX NG kann auch als musterbasiert eingestuft werden.
[2] RTG bei Verwendung von Wildcards.
[3] DTD ist Teil der XML Recommendation, XSD wird in eigenständigen Spezifikationen definiert.
[4] Kontrolle der Struktur durch Regeln möglich.
[5] Einbindung externer Datentypen-Bibliotheken möglich.
[6] Kontrolle der Datentypen durch Regeln möglich.
[7] Mittels `ID/IDREF/IDREFS`-Token-Typ-Attributen. Für eine formale Untersuchung von Integritäts-Constraints in XML-DTD vgl. Fan und Libkin (2002).
[8] Mittels Datentypen `xs:ID`/`xs:IDREF`/`xs:IDREFS` und `xs:key`/`xs:keyref`, `xs:unique`. Für eine formale Untersuchung von Integritäts-Constraints in XSD vgl. Arenas *et al.* (2002).
[9] Mittels `ID/IDREF/IDREFS`-Token-Typ-Attributen bei Verwendung des DTD-Kompatibilitätsmodus (verbunden mit Einschränkungen der formalen Ausdrucksstärke).
[10] Kontrolle der Integrität durch Regeln möglich.
[11] Im Sinne von 1-nicht-ambig; in XSD durch UPA forciert.
[12] Bei SCHEMATRON ist die Erstellung nicht-deterministischer Inhaltsmodelle prinzipiell durch Regeln – in Abhängigkeit von der Host-Schemasprache – möglich.
[13] Nutzung durch Standard-XSLT-Prozessoren möglich.

Mit Ausnahme der RTG sind die Inhaltsmodelle der genannten Grammatikklassen nicht ambig, was die Verarbeitung erleichtert. Zusätzlich wurde mit der Unambigious Restrained Competition Grammar eine zusätzliche Klasse eingeführt, die gut geeignet ist, einen großen Anteil konkreter, d. h., aktiv genutzter RELAX NG-Grammatiken abzubilden.

Tabelle 3.3 fasst die Eigenschaften der diskutierten Constraint Languages (unter zusätzlicher Berücksichtigung von SCHEMATRON) zusammen. Die Darstellung basiert auf den Ausführungen in van der Vlist (2003c, S. 378ff.) und wurde durch die in diesem Kapitel vorgestellten entsprechenden Ergebnisse ergänzt.

3.5 Ebenen und Schichten

Neben den genannten drei Säulen einer Auszeichnungssprache (vgl. die Tripod-Metapher zu Beginn dieses Kapitels) kann bei noch eine weitere Komponenten untersucht werden: Die Möglichkeit der Unterscheidung zwischen der konzeptuellen, logischen Ebene und der technischen Repräsentation in Form einer konkreten Annotation bzw. Serialisierung (Bayerl *et al.* 2003). Während diese Unterscheidung bei einfach annotierten Daten vernachlässigbar ist, kann bei mehrfach annotierten Informationen eine solche Differenzierung sehr sinnvoll sein, um deutlich zu machen, wo die Ursprünge der Annotationen liegen bzw. wodurch es zu Problemfällen wie Überlappungen kommen kann. Witt (2004) unterscheidet hierzu die beiden Begriffe *Level* und *Layer*:

> To avoid confusion when talking about multiply structured text and text ideally organized by multiple hierarchies, the terms „level“ or „level of description“ is used when referring to a logical unit, e. g. visual document structure or logical text structure. When referring to a structure organizing the text technically in a hierarchically ordered way the terms „layer“ or „tier“ are used. A level can be expressed by means of one or more layers and a layer can include markup information on one or more levels. (Witt 2004).

Eine mehr oder minder adäquate deutsche Übersetzung der beiden Begriffe stellen die korrespondierenden Termini konzeptuelle Ebene (*Level*) bzw. Annotationsschicht (oder konkrete Serialisierung, *Layer*) dar, wobei letztere natürlich abhängig von der Notation (der Syntax) ist. Level und Layer können in einer 1:1-, 1:*n*-, *n*:1- oder *n*:*m*-Relation zueinander stehen, wie in der Abbildung 3.15 (nach Goecke, Lüngen *et al.* 2010) ersichtlich wird.

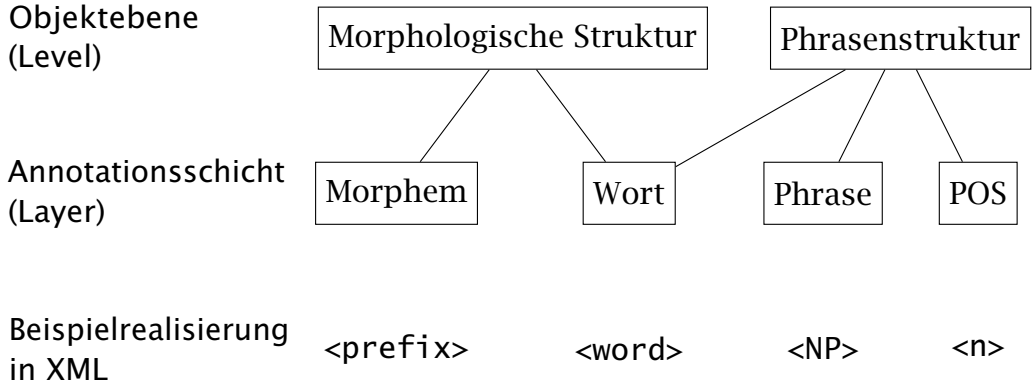


Abbildung 3.15: Relation von Ebene und Annotationsschicht

In einer 1:1-Relation korrespondiert jede konzeptuelle Ebene mit einer Annotationsschicht (also einer konkreten Annotation), z. B., in Form einer XML-Instanz. Bei einer 1:*n*-Relation wird eine einzelne Ebene durch verschiedene XML-Instanzen repräsentiert, beispielsweise, wenn für jede Wortklasse einer POS-Ebene einzelne XML-Instanzen erzeugt werden oder mehrere konkrete Layer für die POS-Ebene vorliegen, weil verschiedene POS-Tagger genutzt wurden, die jeweils eigene XML-Instanzen erzeugen.

Im Falle einer n:1-Relation werden verschiedene Beschreibungsebenen in einer XML-Datei zusammengefasst. Das ist bei nicht-überlappenden Ebenen wie z. B. Syntax und Morphologie möglich und oft praktiziert. Die n:m-Relation schließlich, bei der eine Vielzahl von konzeptuellen Ebenen mit einer Anzahl von konkreten Annotationsschichten kombiniert wird, ist eher selten anzutreffen (vgl. Goecke, Lüngen *et al.* 2010).

Im Verlauf dieser Arbeit kann diese Unterscheidung als ein zusätzliches Kriterium zur Kategorisierung von Auszeichnungssprachen zur strukturierten Speicherung linguistischer Daten angesehen werden, da die Trennung von Level und Layer einen wesentlichen Baustein zur Verwendung generischer Formate darstellt und darüber hinaus auch multiple Annotationshierarchien erleichtert.

3.6 Zusammenfassung

Wenn in den folgenden Teilen dieser Arbeit Auszeichnungssprachen für die Strukturierung linguistischer Daten behandelt werden, dann lassen sich diese immer unter verschiedenen Gesichtspunkten charakterisieren: Syntax und Notation (erstere vorgegeben durch die Metasprache, entweder XML oder nicht-XML), formales Modell (Baum oder Graph), Grammatik, und Unterscheidung zwischen Konzept und Serialisierung.

XML-basierte Auszeichnungssprachen haben prinzipiell den Vorteil, dass sie eine hervorragende Unterstützung durch Softwareprodukte genießen und darüber hinaus mittels der weiteren Spezifikationen aus dem Umfeld des *W3C* wie XPath, XSLT, XQuery sehr flexibel genutzt werden können. In der Frage der Notation lässt sich die Inline- von der Standoff-Notation unterscheiden.

Das Datenmodell des Baumes ist nicht zwingend die einzige Möglichkeit in XML-kodierte Informationen zu formalisieren. Der in diesem Kapitel neu eingeführte erweiterte Baum (xT) ist – z. B. durch Nutzung des Integritäts-Mechanismus – neben anderen Graphen-basierten Strukturen eine Alternative. Nicht auf XML basierende Metasprachen sind zwar generell nicht auf das formale Modell des Baums beschränkt, leiden aber unter einer nur marginal vorhandenen Softwareunterstützung.

Für die Grammatik sind Einschränkungen und Unterschiede in der formalen Ausdrucksstärke eher theoretischer Natur. Obwohl RELAX NG ein formal mächtiger Grammatikformalismus ist und der vollen Ausdrucksstärke einer regulären Baumgrammatik (RTG) entspricht, reichen in der praktischen Verwendung entweder die in diesem Abschnitt neu eingeführten URCGs oder – aufgrund von Kompatibilität zu den beiden anderen weit verbreiteten Formalismen XML Schema und DTD – STG oder LTG aus, um einen Großteil an produktiv eingesetzten Schemata abzubilden. Dennoch darf die Wahl der zu Grunde liegenden XML-Schemasprache bei der Betrachtung von Annotationsformaten nicht außer Acht gelassen werden, da auch funktionale Unterschiede ausschlaggebend sein können.

Bei Auftreten multipler Auszeichnungen über ein und dasselbe Primärdatum ist zusätzlich noch ein Augenmerk auf die Unterscheidung zwischen Level und Layer zu richten. Auszeichnungssprachen, die prinzipiell mehrfach annotierte Daten zulassen, können – gerade bei einer größeren Anzahl an Annotationsebenen – durch die Unterstützung einer solchen Differenzierung Relationen zwischen Elementen verschiedener

Level von denen verschiedener Layer abgrenzen und damit eine Auswertung relevanter Beziehungen spürbar erleichtern.

Eine Auszeichnungssprache mit möglichst hohem Nutzwert sollte daher XML-basiert sein, über das formale Modell eines Baumes hinaus Unterstützung für äußere Kanten erlauben, und einen Grammatikformalismus nutzen, der entsprechend mächtig genug ist, linguistisch relevante Phänomene abbilden lassen zu können.[71] Eine darüber hinaus gehende Möglichkeit der Unterscheidung zwischen der konzeptuellen Ebene einer Annotation und deren Serialisierung vervollständigt den Kriterienkatalog. Die in den folgenden Abschnitten vorgestellten Ansätze werden auf diese Ansprüche hin überprüft werden.

[71] Zu beachten ist, dass an dieser Stelle noch keine Aussage über das Inventar der Grammatik, also die zur Verfügung stehenden Einheiten der Informationsstrukturierung, getroffen wird.

Teil II

Maßgebliche Standards und Normen

The primary conclusion is that XML-based language design is an activity best avoided; not only because it is expensive and dangerous, but because it may in many cases be actively harmful. If there is an alternative, usually an existing language which can plausibly be repurposed, that alternative should be given a chance if at all possible.

Bray (2005, S. 7)

4 Einleitung

Gegenstand dieses Teils der Arbeit sind Spezifikationen zur Strukturierung linguistischer Korpora. Ein wesentlicher Aspekt standardisierter Formate ist deren Nachhaltigkeit, so dass sie im Sinne des oben angeführten Zitats vor der Entwicklung eines neuen Annotationsformats konsultiert werden sollten.

In den folgenden Kapiteln werden sowohl internationale Normen wie das LINGUISTIC ANNOTATION FRAMEWORK (Kapitel 9), aber auch De-Facto-Standards wie die TEI GUIDELINES (Kapitel 5) oder der CORPUS ENCODING STANDARD (Kapitel 7) thematisiert. Dass beide Gebiete nicht immer klar voneinander zu trennen sind, zeigen die in Kapitel 6 diskutierten standardisierten Beschreibungen von Merkmalsstrukturen: Sie wurden im Rahmen der TEXT ENCODING INITIATIVE entwickelt und dann als internationale Normen ISO 24610-1:2006; ISO 24610-2:2011 veröffentlicht. Dagegen sind das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK (Kapitel 10) und das damit eng verwandte SYNTACTIC ANNOTATION FRAMEWORK (Kapitel 11) unmittelbare Ergebnisse der Normierungsarbeit (die natürlich auch äußeren Einflüssen unterliegt, vgl. Abschnitt 2.1.1). Diese Normen wurden (und werden) maßgeblich im Rahmen des ISO/TC 37/SC 4 „Language Resouce Management“ und dessen nationalen Spiegelgremien (in Deutschland der NA 105-00-06 AA „Sprachressourcen“) entwickelt bzw. begutachtet. Das Unterkomitee 4 des ISO/TC 37 wurde 2001 eingerichtet, um Prinzipien und Methoden zur Erstellung und Verarbeitung linguistischer Ressourcen zu entwickeln. Das Sekretariat übt das *Korea Terminology Research Center* (*KORTEM*) als Vertreter des nationalen koreanischen Standardisierungsinstitut *Korean Agency for Technology and Standards* (*KATS*) aus.

Eine besondere Rolle unter den in dieser Arbeit diskutierten Spezifikationen nimmt der Internationale Standard DATA CATEGORY REGISTRY (Kapitel 8) ein: Diese Spezifikation beschreibt nicht ein formales Modell oder Tagset zur Annotation, sondern formuliert Methoden und Prozesse für den Aufbau und die Pflege eines zentralen Registers konkreter Begriffe und damit verknüpfter Konzepte, an denen sich Annotationsinventarien oder Metadaten orientieren können. Diese Datenkategorienregistrierung wird beispielsweise von den Spezifikationen MORPHO-SYNTACTIC ANNOTATION FRAMEWORK

(Kapitel 10) und dem damit eng verwandten SYNTACTIC ANNOTATION FRAMEWORK (Kapitel 11) verwendet und wurde im Rahmen des ISO/TC 37/SC 3 „Systems to manage terminology, knowledge and content“ entwickelt. Dieses Unterkomitee wurde bereits 1989 eingerichtet, um internationale Normen im Bereich computer-unterstützte Terminologie zu erarbeiten. Das *Deutsche Institut für Normung* (DIN) übt das Sekretariat von ISO/TC 37/SC 3 aus.

Eine abschließende Übersicht über die Gesamtstruktur des TC 37 vermittelt folgende Auflistung:

- ISO/TC 37/SC 01 „Principles and methods“
  - WG 02 „Harmonization of terminology“
  - WG 03 „Principles, methods and vocabulary“
  - WG 04 „Socioterminology“
  - WG 05 „Concept modelling in terminology work“
- ISO/TC 37/SC 02 „Terminographical and lexicographical working methods“
  - WG 01 „Language coding“
  - WG 02 „Terminography“
  - WG 03 „Lexicography“
  - WG 04 „Source identification for language resources“ – z. Zt. ruhend
  - WG 05 „Requirements and certification schemes for cultural diversity management“
  - WG 06 „Translation and interpretation processes“
- ISO/TC 37/SC 03 „Systems to manage terminology, knowledge and content“
  - WG 01 „Data elements“
  - WG 02 „Vocabulary“
  - WG 03 „Data interchange“
  - WG 04 „Database management“
- ISO/TC 37/SC 04 „Language resource management“
  - WG 01 „Basic descriptors and mechanisms for language resources“
  - WG 02 „Representation schemes“
  - WG 03 „Multilingual information representation“
  - WG 04 „Lexical resources“
  - WG 05 „Workflow of language resource management“
  - WG 06 „Linguistic annotation“
- ISO/TC 37/TC 00G „Terminology Coordination Group for TC 37“

- ISO/TC 37/WG 08 „Ontologies – Linguistic, terminological and knowledge organization aspects“
- ISO/TC 37/WG 09 „Data category registry“

Es ist davon auszugehen, dass nach Beendigung des Standardisierungsprozesses (diese Phase ist bei einigen Spezifikationen bereits abgeschlossen) zumindest einige der in diesen Komitees und Arbeitsgruppen diskutierten Normen Eingang in die wissenschaftliche Praxis finden werden.

4.1 Nicht berücksichtigte Internationale Standards

Sowohl das Lexical Markup Framework (LMF, ISO 24613:2008) als auch das Semantic Annotation Framework (SemAF, bestehend aus ISO/DIS 24617-1; ISO/AWI 24617-5; ISO/AWI 24617-4; ISO/DIS 24617-2) werden nicht weiter im Rahmen dieser Arbeit diskutiert. LMF kann durchaus auch für die Annotation linguistischer Korpora relevant sein, da der Gegenstand der Spezifikation eine linguistische Repräsentation lexikalischer Informationen und deren Einsatz in NLP-Anwendungen ist (Francopoulo *et al.* 2006; Francopoulo *et al.* 2009).[1] Maschinenlesbare Lexika sind nicht das primäre Ziel des Standards, denkbar wäre ein Einsatz von LMF als Lexikonrepräsentation im Rahmen des Morpho-syntactic Annotation Framework (Kapitel 10) oder ähnlicher generischer Formate. Als weiterer Standards zur Strukturierung lexikalischer Informationen wäre ISO 1951:2007 zu nennen, ein Vergleich dieser Spezifikation zum entsprechenden Kapitel der TEI Guidelines (P5 1.9.1, Kapitel 9, „Dictionaries“) findet sich in Lemnitzer, Romary *et al.* 2012. Da die genannten Spezifikationen allerdings nicht im direkten Fokus dieser Arbeit liegen, werden sie nicht weiter behandelt.

Die aus fünf Einzelspezifikationen bestehende Norm SemAF ist aktuell in der Entwicklung nicht sehr weit fortgeschritten. Zwar sind ISO/DIS 24617-1 und ISO/DIS 24617-2 im Status eines DIS veröffentlicht, die weiteren Standards stehen allerdings noch am Anfang der Normungsarbeit.[2]

4.2 PAULA

Ebenfalls nicht ausführlich vorgestellt wird das Potsdamer Austauschformat für linguistische Annotationen (PAULA), das 2005 in Potsdam im Projekt „D1: Linguistic Database: Annotation and Retrieval“ des SFB 632 entwickelt wurde und als generisches Austauschformat starke Ähnlichkeit zu GrAF (Abschnitt 9.2) aufweist (vgl. Dipper und Götze 2005). Konzeptuell nimmt PAULA Anleihen beim Annotation Graph sowie NITE (vgl. Abschnitt 3.3.3.2), das formale Modell, PAULA Object Model (POM) genannt, entspricht dem eines gelabelten DAG (Stede und Bieler 2011, S. 20). Der Aufbau des Formats folgt dem von Ide, Kilgarriff *et al.* 2000; Ide und Romary 2001b propagierten

[1] Eine Diskussion des ISO 24613:2008 für den Bereich etymologischer Daten findet sich in Alt (2006).
[2] Der dritte Teil ist zum Zeitpunkt dieser Arbeit noch nicht mit einer Projektnummer bei der ISO registriert.

Generic Mapping Tool (GMT), das als Teil der internationalen Norm Terminologcial Markup Framework (TMF, ISO 16642:2003) standardisiert und in einer früheren Fassung des Internationalen Standards ISO 12620:2009 (ISO/CD 12620-1) enthalten ist. Es verwendet ein generisches Tagset, das die Elemente `mark` (zur Kennzeichnung von *Markables*, den basalen Einheiten der Auszeichnung), `feat` (*Feature*, Merkmal) und `struct` (*Structure*) beinhaltet, um den Typ einer entsprechenden Annotationsebene (Layer im Sinne von Abschnitt 3.5) auszuzeichnen. Diese Ebenen werden in separaten Dateien gespeichert, ebenso wie das Primärdatum, auf das die Annotation(en) bezogen werden (Dipper 2005; Dipper und Götze 2005, S. 207). Dabei werden XML-inhärente Strukturen, wie hierarchische Beziehungen zwischen Mutter- und Tochterknoten durch explizite Annotation ersetzt (z. B. durch Verwendung einer *Structure List*). Vorteil dieser Verarbeitung ist die spätere Möglichkeit der standardisierten Analyse einer Vielzahl von Annotationen (da die strukturellen Beziehungen durch generisches Markup in der Structure List ersichtlich sind). Als Constraint Language kommt XML-DTD zum Einsatz (Dipper, Götze, Küssner *et al.* 2007, S. 339, Listing 1).

Bird, Y. Chen *et al.* (2006); Dipper, Götze, Küssner *et al.* (2007) untersuchen anhand exemplarischer Abfragen (erstere mit der erweiterten XPath-Notation LPath, letztere mittels XQuery) die Verarbeitungszeiten von PAULA bei linguistischen Korpora. Dabei kommen Dipper, Götze, Küssner *et al.* (2007, S. 343f.) zu dem Schluss, dass die Standoff-Annotation des PAULA-Formats in Bezug auf die Verarbeitungszeiten nicht effizient genug ist: „With large corpora, execution times for the standoff version are unacceptably slow. [...] [I]t does not support efficient processing“.

Die Autoren empfehlen anschließend die Transformation in eine Inline-Annotation mit *Fragmenten* (vgl. Abschnitt 3.3.2.2) zur Kodierung überlappender Strukturen – wodurch das eigentlich primäre Standoff-Notationskonzept etwas konterkariert wird. Positiv zu bewerten ist der Einsatz als Import- und Export-Format der ANNIS-Datenbank (Dipper, Götze, Stede *et al.* 2004; Götze und Dipper 2006). Da es sich allerdings weder um einen Internationalen Standard noch um eine weit verbreitete Spezifikation handelt, werden in dieser Arbeit keine weiteren Aussagen zu PAULA gemacht.

4.3 DocBook

Für die strukturierte Speicherung linguistischer Daten nicht primär ausgelegt und daher in dieser Arbeit ebenfalls nicht ausführlich berücksichtigt wird die internationale Spezifikation DocBook (Walsh und Muellner 1999). DocBook dürfte neben der TEI (vgl. Kapitel 5) die am Häufigsten genutzte XML-Spezifikation zur Strukturierung textueller Informationen sein. Obwohl DocBook eher zur Gestaltung von Büchern, Artikeln und vorrangig für den Einsatz in der technischen Dokumentation konzipiert wurde, ist nicht nur dessen Einfluss auf linguistisch motivierte Spezifikationen sehr groß; DocBook wird auch selbst zur Speicherung von Korpusdaten herangezogen, so etwa in den Projekten „Generische Dokumentstrukturen in linear organisierten Texten“ (SemDok) der Universität Gießen und „Hypertextualisierung auf textgrammatischer Grundlage“

(HyTex) der Universität Dortmund.[3]

DocBook wurde 1991 durch *HaL Computer Systems* und *O'Reilly* konzipiert, mit dem Ziel, den Austausch an UNIX-Dokumentationen zu standardisieren, und wird auch heute noch stetig weiter entwickelt. Seit 1998 wird DocBook als Technical Commitee (TC) der *Organization for the Advancement of Structured Information Standards (OASIS)* geführt, in den Jahren 1994 bis 1998 wurde die Pflege von der *Davenport Group* übernommen (Schraitle 2004). Aktuell ist die offizielle Versionsnummer 5.0, die die Version 4.5 vom 03. Oktober 2006 ablöst, und von der einige größere Veränderungen ausgehen.[4] So werden verschiedene Elemente wie `bookinfo` und `articleinfo` zum generischen Element `info` zusammengefasst und globale XML-Attribute wie `xml:id` ersetzen interne und lokal deklarierte `id`-Attribute. Mit `http://docbook.org/ns/docbook` wurde ein neuer Namensraum definiert und RELAX NG als primäre Constraint Language etabliert.[5] Die Änderungen zwischen den Versionen 4.x und 5.0 werden detailliert in Kosek *et al.* (2006) beschrieben.

DocBook ist als SGML- und XML-Auszeichnungssprache frei erhältlich und wird auch von einigen Textverarbeitungen unterstützt. So ist es zum Beispiel in der Open-Source Office-Suite OpenOffice bzw. LibreOffice möglich, aus der Textverarbeitung *Writer* heraus DocBook-konforme Dokumente zu erzeugen.[6] Mit Hilfe von DocBook lässt sich eine Reihe von Texttypen strukturieren, darunter gedruckte Bücher, Websites, Technische Dokumentationen (unter anderem der Projekte „XFree86“, „GNOME“, „KDE“ und „FreeBSD“), aber auch Spezifikationen (vgl. Pawson 2005).

Prinzipiell kann DocBook als sehr gut unterstützte Quelle für Single-Source-Publishing-Anwendungen dienen, da neben den DocBook-Schemata auch eine Reihe an XSLT-Stylesheets zur Verfügung steht, die eine Reihe von Ausgabeformaten generieren (neben HTML auch Unix Man Pages, LaTeX, XSL-FO oder PostScript). Schraitle (2004, S. 86) führt folgende Vorteile von DocBook an:

- Die Bedeutung DocBooks als Quasi-Industrie-Standard aufgrund seiner hohen Verbreitung
- Die Anpassungsmöglichkeiten in Bezug auf den modularen Aufbau
- Die Freiheit, DocBook ohne Zahlung von Lizenzgebühren nutzen zu können[7]

[3] Zu beachten ist, dass DocBook hier jeweils nur für die Annotationsebene der logischen Dokumentstruktur herangezogen wird. Nähere Angaben zum Projekt SemDok finden sich auf `http://www.uni-giessen.de/cms/fbz/fb05/germanistik/iprof/asclhome/forschung/semdok`, zum Projekt HyTex unter `http://www.hytex.tu-dortmund.de/`. Jeweils zuletzt abgerufen am 19.04.2012.

[4] Parallel dazu ist DocBook 5.1 in Form einer Beta-Version erhältlich.

[5] Die Implementierung in Form eines RELAX-NG-Schemas (inkl. eingebetteter Schematron-Regeln) gilt als normativ, die Schema-Implementierungen in Form einer DTD und eines XSD haben informativen Charakter und können in Bezug auf die in RELAX NG und Schematron gemachten Constraints abweichen (für Aussagen zur Ausdrucksmächtigkeit der verschiedenen Constraint Languages sei auf Abschnitt 3.4 verwiesen).

[6] In den aktuellen Versionen OpenOffice 3.3 bzw. LibreOffice 3.4 werden dabei DocBook-Instanzen in der Version 4.1.2 generiert.

[7] Die Spezifikation steht in Form verschiedener Schemaformate zum kostenlosen Download auf `http://www.docbook.org` zur Verfügung, zuletzt abgerufen am 19.04.20121.

- Die Investionssicherheit bedingt durch die langen Innovationszyklen
- Die gute Unterstützung durch Open-Source-Software

Insgesamt enthält DocBook 5.0 über 390 Elemente zur strukturellen und logischen Auszeichnung von Texten und bindet über XML Namespaces weitere Auszeichnungssprachen (darunter auch MathML und SVG) direkt ein. Ein Nachteil von DocBook ist das Fehlen von linguistisch motivierten Annotationseinheiten. So ist es zwar möglich, über XML Namespaces spezifische Auszeichnungssprachen für diesen Zweck einzubinden, allerdings kann es dadurch gegebenenfalls zu Überlappungen (vgl. Abschnitt 3.3.2) kommen. Auch die Möglichkeit mit Hilfe des `set`-Elements (zur Gruppierung von Büchern) einen Korpus linguistischer Daten abzubilden, ist wenig erfolgsversprechend, da diese Vorgehensweise das angesprochene Grundproblem nicht löst.

Rahtz *et al.* (2004) zeigen sowohl allgemeine Überschneidungen zwischen DocBook und den in Kapitel 5 diskutierten TEI Guidelines auf, als auch konkrete Möglichkeiten Konstrukte aus beiden Tagsets miteinander zu kombinieren. Dieser Ansatz wäre eine viel versprechende Lösung, sofern eine der beiden Spezifikationen aus vorliegenden Gründen zwingend zur Strukturierung bestimmter Aspekte vorgeschrieben ist. Fehlen solche externen Zwänge sollte dagegen eine der in den folgenden Kapiteln dieser Arbeit diskutierten Spezifikationen zur Annotation linguistischer Daten eingesetzt werden. Diese werden anhand der in Kapitel 3 aufgestellten vier Komponenten von Auszeichnungssprachen (Syntax/Notation, formales Modell, Dokumentgrammatik und Möglichkeit der Trennung von Konzept und Notation) beurteilt.

If asked for a sure recipe for chaos I would propose a project in which several thousand impassioned specialists in scores of disciplines from a dozen of more countries would be given five years to produce some 1300 pages of guidelines for representing the information models of their specialities in a rigorous, machine-verifiable notification.

Goldfarb (1998, S. 1)

5 Die Text Encoding Initiative

Die Terme „Text Encoding Inititative“ bzw. „TEI“ haben drei Lesarten: (1) die Bezeichnung einer Organisation (*TEI Consortium*), (2) die von dieser herausgegebenen Spezifikation (TEI GUIDELINES), und nicht zuletzt (3) die Bezeichnung einer weltweit agierenden Gemeinschaft von Wissenschaftlern, die sich mit der Strukturierung von textuellen Informationen befasst (vgl. Jannidis 2009). Die TEI GUIDELINES sind gut zehn Jahre älter als XML selbst und stellen zusammen mit DOCBOOK (vgl. Abschnitt 4.3) die dienstälteste Spezifikation zur Strukturierung textueller Informationen dar. Sie besteht aus einem umfangreichen Annotationsinventar nebst Dokumentation. Das Tagset ist eine komplexe SGML- bzw. XML-Anwendung, deren modularer Aufbau eine Reihe von Formaten umfasst, die zur Annotation einer ganzen Bandbreite von Ressourcen genutzt werden können. Bis heute ist die Spezifikation die einzige, die dem Anspruch gerecht wird, ein vollständiges Modell für die Annotation von Texten jeglicher Art und jeglicher Sprache anzubieten.[1]

Dabei vertreten die TEI GUIDELINES nicht den Anspruch ein Standard zur Auszeichnung linguistischer Daten zu sein. Die Bezeichnung „Guidelines“ ist gewusst gewählt: Zum einen soll die intellektuelle Freiheit des Forschenden nicht unnötig eingeschränkt werden, zum anderen sollen dennoch sinnvolle Vorgaben zur Bezeichnung (und entsprechend: Annotation) linguistischer Phänomene gemacht werden, um eine Austauschbarkeit der so ausgezeichneten Daten zu gewährleisten. Neben der Tatsache, dass ein Großteil der in den TEI GUIDELINES deklarierten Merkmale optional ist und oftmals mehrere Annotationsvarianten zur Verfügung stehen, ist das Tagset auch jederzeit erweiterbar – wobei die Grenze zwischen konformen (*TEI Conformant*) und nichtkonformen Dokumenten genau festgelegt ist (Ide und Sperberg-McQueen 1995, S. 8; P5 1.9.1, Abschnitt 23.3, „Conformance“). Ebenso kann unterschieden werden zwischen

[1] Die Betonung liegt hier auf „Texte“; wie Wörner (2009, S. 64) ausführt, konnte sich die TEI „für Korpora gesprochener Sprache nie richtig durchsetzen“. Daher werden in diesem Kapitel Mechanismen zur Alignierung und Elemente zur zeit-basierten Annotation wie das `timeline`-Element aus P5 1.9.1, Abschnitt 16.5, „Synchronization“ nicht weiter berücksichtigt.

der Annotation eines Textes zur lokalen Verwendung mittels eines angepassten TEI-Tagsets (oder gar einer komplett davon unabhängigen Auszeichnungssprache) und der Verwendung TEI-konformer Instanzen zum Zwecke des Datenaustauschs zwischen verschiedenen Wissenschaftlern – gerade letztere Möglichkeit ist einer der Grundgedanken bei der Entwicklung der Spezifikation gewesen.

5.1 Historie

Am 11. und 12. November 1987 traf sich eine Gruppe von 32 Wissenschaftlern auf einer von der *Association for Computers and the Humanities* (*ACL*) einberufenen und von der *U.S. National Endowment for the Humanities* (Stiftung für Geisteswissenschaften) geförderten Versammlung im *Vassar College* in Poughkeepsie, New York (vgl. Ide und Sperberg-McQueen 1995; Mylonas und Renear 1999). Die Gruppe war äußerst heterogen zusammen gesetzt: Neben Vertretern verschiedener wissenschaftlicher Fachrichtungen waren auch Bibliotheken und Archive sowie einige Projekte vertreten – dazu kam, dass die Teilnehmer sowohl aus einer Reihe europäischer Länder als auch aus Nordamerika und Asien stammten. Ziel des Treffens war die Identifikation eines Problems, das es zu lösen galt: die Vereinheitlichung von digital gespeicherten Texten. Zwischen den 1960er und ’80er Jahren wurden mehr und mehr proprietäre Formate und Methoden entwickelt, um wissenschaftlich relevante Texte digital zu verarbeiten, da die bis dahin etablierten Textverarbeitungssysteme alle zu allgemein und zu ungenau waren. Der Großteil dieser Formate war untereinander inkompatibel und auch erste Versuche Ende der 1970er und Anfang der ’80er Jahre, Kodierungsstandards für digital zu verarbeitende wissenschaftliche Texte zu verabschieden, scheiterten (Ide und Sperberg-McQueen 1995, S. 5). Beim besagten Treffen im November 1987 einigte man sich schließlich auf einige grundlegende Richtlinien, die später als „Poughkeepsie Principles“ bekannt wurden:[2]

1. The guidelines are intended to provide a standard format for data interchange in humanities research.
2. The guidelines are also intended to suggest principles for the encoding of texts in the same format.
3. The guidelines should
   a) define a recommended syntax for the format,
   b) define a metalanguage for the description of text-encoding schemes,
   c) describe the new format and representative existing schemes both in that metalanguage and in prose.
4. The guidelines should propose sets of coding conventions suited for various applications.
5. The guidelines should include a minimal set of conventions for encoding new texts in the format.

[2] Das Protokoll des Schlussworts mit den Richtlinien ist auch weiterhin einsehbar unter der URL `http://www.tei-c.org/Vault/SC/teipcp1.txt`, zuletzt abgerufen am 19.04.2012.

6. The guidelines are to be drafted by committees on
   a) text documentation
   b) text representation
   c) text interpretation and analysis
   d) metalanguage definition and description of existing and proposed schemes,

   coordinated by a steering committee of representatives of the principal sponsoring organizations.
7. Compatibility with existing standards will be maintained as far as possible.
8. A number of large text archives have agreed in principle to support the guidelines in their function as an interchange format. We encourage funding agencies to support development of tools to facilitate this interchange.
9. Conversion of existing machine-readable texts to the new format involves the translation of their conventions into the syntax of the new format. No requirements will be made for the addition of information not already coded in the texts.

Rückblickend kann man sagen, dass vor allem die Festlegung auf die ein Jahr zuvor als ISO-Standard verabschiedete Metasprache SGML in Kombination mit der Dringlichkeit des Problems und der Zusammensetzung der Gruppe den späteren Erfolg der TEI GUIDELINES ermöglichten. Noch im gleichen Jahr wurden durch die *Association for Computers and the Humanities* (ACH), der *Association for Computational Linguistics* (*ACL*) und der *Association for Literary and Linguistic Computing* (*ALLC*) die *Text Encoding Initiative* als internationales und multi-linguales Projekt ins Leben gerufen. Unterstützt wurde das junge Gremium unter anderem weiterhin von der *US National Endowment for the Humanities*, der *Europäischen Union* (Direktorat XIII, CEC/DG-XIII), dem *Social Science and Humanities Research Council of Canda* und der *Andrew W. Mellon Foundation*.

Die erste Version der TEI GUIDELINES die P1 („Proposal 1“) wurde im Juli 1990, knapp drei Jahre nach Gründung der Initiative, veröffentlicht. Zu diesem Zeitpunkt gab es zwei leitende Redakteure (einen Chefredakteur und einen Mitherausgeber), vier Arbeitsgruppen und weit über 50 teilnehmende Wissenschaftler (Mylonas und Renear 1999, S. 4). Aus den Arbeitsgruppen wurden in der Anfangszeit vier Komitees: eines für Textdokumentation (*Committee for Text Documentation*), eines für Textrepräsentation (*Committee for Text Representation*), eines für Textanalyse und -interpretation (*Commitee on Text Analysis and Interpretation*) und für die syntaktische Fragen (*Commitee on Metalanguage Issues*, vgl. Ide und Sperberg-McQueen 1995, S. 11ff. und die obige Aufstellung der „Poughkeepsie Principles“). Im November 1990 folgte eine korrigierte Version 1.1. In der anschließenden zweiten Phase, mit nunmehr 15 Arbeitsgruppen, wurden zwischen 1990 und 1993 zusätzliche Annotationsempfehlungen erarbeitet und

in Form von Einzelkapiteln veröffentlicht (P2).[3] Die erste offizielle vollständige Fassung der TEI GUIDELINES wurde im Mai 1994 als P3 und 1999 als P4-BETA veröffentlicht. Seit dieser Fassung ist die TEI sowohl in digitaler als auch gedruckter Form erhältlich.

Die Version P4 wurde im Juni 2002 veröffentlicht und nutzt erstmalig XML als zu Grunde liegende Metasprache und ist im Wesentlichen eine Reformulierung der P3. Unter anderem stellt sie einige DTD-Fragmente zur Verfügung, die sowohl von der SGML- als auch der XML-Version genutzt werden können und daher eine gewisse Abwärtskompatibilität herstellen: Jedes P3-Dokument sollte eine konforme P4-Instanz sein, vorrangig wurden daher Fehler in der Spezifikation korrigiert. Bauman, Bia *et al.* (2004) präsentieren eine Reihe von Empfehlungen zur Migration P3-annotierter linguistischer Daten in eine P4-konforme Fassung am Beispiel des „British National Corpus“ (BNC). Aufgrund dieser eher technischen Unterschiede blieb die P4 über einen langen Zeitraum quasi unverändert, weshalb zeitgleich die Arbeiten an der aktuellen Version der TEI, P5, begannen, die einige wesentliche Änderungen mit sich bringt (u. a. die Nutzung von RELAX NG als Dokumentgrammatikformalismus); die Arbeiten an dieser Version sind dokumentiert in Burnard (2007); Wittern *et al.* (2009). Sowohl die Entwicklungsversionen der TEI GUIDELINES als auch zugehörige Programmen werden seitdem unter der *GNU General Public License* (GPL) auf der Plattform SOURCEFORGE zugänglich gemacht.[4]

Seit dem 1. November 2007 ist die P5 1.0.0 offiziell verfügbar (zunächst online, später auch in gedruckter Form), die P4 wird nach Auskunft der Herausgeber aufgrund der teilweise deutlichen konzeptuellen Unterschiede noch bis November 2012 unterstützt. Die Grundlagen für die in die aktuelle Version eingeflossenen Änderungen wurden in den einzelnen Arbeitsgruppen (*Special Interest Group*, SIG) gelegt. Die P5 enthält daher eine Reihe neuer Elemente und bringt Änderungen mit, die die bisher sichergestellte Abwärtskompatibilität brechen (z. B. durch Reformulierungen einiger Inhaltsmodelle). Darüber hinaus wurde ein Versionierungssystem (bestehend aus *Major Number* und *Minor Number*) eingeführt, um abwärtskompatible Neuerung von nicht-kompatiblen Änderungen zukünftig unterscheiden zu können.

Bereits im Januar 1999 präsentierten die Universitäten von Virginia und Bergen dem Exekutivkomitee der TEI den Vorschlag, ein internationales Konsortium zu gründen. Die Annahme des Vorschlags resultierte in der Gründung des *TEI Consortiums*, das seit Dezember 2000 als institutioneller Rahmen und Ansprechpartner für alle Belange der Spezifikation dient. Als wissenschaftlich und wirtschaftlich unabhängige und gemeinnützige Organisation trägt das *TEI Consortium* wesentlich zur Verbreitung des TEI-Tagsets und der weiteren Entwicklung bei.

[3] Die Versionen der TEI enthalten üblicherweise eine Übersicht über bisherige Ausgaben. In dieser wird für die P2 das Jahr 1992 angegeben. Es liegen allerdings interne Versionen vor, die bis weit in das Jahr 1993 hinein datiert sind und dennoch als P2 bezeichnet werden. Ähnlich verhält es sich mit der aktuellen Version P5. Diese wurde erstmalig 2007 veröffentlicht (P5 1.0.0) und wird sukzessive aktualisiert. Hier gibt eine Versionsnummer nähere Auskunft.

[4] Informationen und Downloads unter `http://tei.sf.net/`, zuletzt abgerufen am 19.04.2012. Mit der Dezember 2011 veröffentlichten Version 2.0, die nicht mehr für diese Arbeit berücksichtigt wurde, wurde die Lizenz geändert: Der Großteil des vom *TEI Consortium* herausgegebenen Materials wird seitdem sowohl unter der *Creative Commons Attribution 3.0 Unported License* (CC BY 3.0) als auch einer *BSD 2-Clause license* lizensiert.

5.2 Das TEI-Tagset

Die TEI GUIDELINES definieren ein modular aufgebautes Tagset (d. h., eine XML-Auszeichnungssprache, vormals eine SGML-Anwendung) zur Annotation von Texten aller Art zum Zwecke der wissenschaftlichen Forschung. Stand ursprünglich der Gedanke an ein universelles Austauschformat im Vordergrund, hat sich die Spezifikation seit einigen Jahren zu einem weit verbreiteten Werkzeug zur strukturierten Auszeichnung und dauerhaften Speicherung von textuellen Informationen entwickelt. Die Modularität der Auszeichnungssprache wurde mit der Version P5 stark erweitert und zeigt sich in zweierlei Hinsicht: Zum einen ist das Tagset selbst hoch modular gestaltet, d. h., einem geringen Anteil an obligatorischen Elementen und Attributen steht eine ungleich größere Menge an optionalen Einheiten der Informationsmodellierung gegenüber, die in Modulen, sowie Element- und Attributklassen zusammengefasst sind. Zum anderen verzichtet die TEI seit der Version P5 auf die in früheren Fassungen noch vorhandenen eigenen Implementierungen zu Gunsten von standardisierten Verfahren. Ein Beispiel dafür ist die Neuformulierung der TEI Extended Pointer (vgl. P4, Kapitel 14, „Linking, Segmentation, and Alignment“, S. 352ff.), die in der P5 durch Nutzung von XPOINTER realisiert wurden (vgl. P5 1.9.1, Kapitel 16, „Linking, Segmentation, and Alignment“, S. 494ff.). Daher ist das Tagset nicht nur im Rahmen der innerhalb der TEI definierten Strukturen flexibel adaptierbar, sondern lässt sich darüber hinaus einfach durch fremde Konstrukte erweitern.

Der Nutzer selbst kann aus einer Reihe von Modulen das Annotationsformat erstellen, das seinen Bedürfnissen entspricht (oder ihnen möglichst nahe kommt). Jedes Modul der Spezifikation definiert eine Gruppe von Elementen und Attributen mit deren dazugehörigen Inhaltsmodellen und ergänzt das *core*-Modul, das Informationseinheiten enthält, die in allen TEI-Dokumenten enthalten sind. Daneben sind auch Teile des Moduls *header* mit den Metadaten und des Moduls *textstructure* Teil einer minimalen Dokumentgrammatik zur Strukturierung einer TEI-Instanz, die als „TEI Absolute Bare“ bezeichnet wird.[5] Umfasst die TEI P4 noch 441 Elemente in 80 Elementklassen und 107 Entitäten, enthält die P5 in der vollständigen Fassung (d. h. mit allen Modulen) 520 Elemente und 439 Attribute (unterteilt in 22 Module).[6] Diese Komplexität erleichtert nicht unbedingt den Zugang zu diesem mächtigen Annotationswerkzeug. Aus diesem Grund – und natürlich aus der Tatsache, dass der modulare Aufbau der Spezifikation genau dieses zulässt – wurde bereits 1995 auf Basis der P3 eine vereinfachte Version, die TEI LITE, veröffentlicht (Burnard und Sperberg-McQueen 1995; Burnard und Sperberg-McQueen 2006).[7] Mit 145 Elementen ist sie eine der am meisten genutzten Anpassungen (*Customization*), da sie für den Großteil der Einsatzzwecke alle erforderlichen Einheiten der Informationsstrukturierung beinhaltet. Auch wenn die TEI LITE die

[5] Auch für das Modul *core* gilt, dass nicht alle dessen Elemente und Attribute in dieser Dokumentgrammatik enthalten sind. Insgesamt besteht sie aus nur 18 Elementen und 12 Attributen.

[6] Ergebnis einer Zählung der Vorkommen von `xs:element` mit dem Attribut `name` (also ohne Referenzen) in einem XSD der aktuellen Version P5. Zu beachten ist, dass das Element `egXML` in der XSD-Fassung in das externe Schema-Fragment `example.xsd` ausgelagert wurde, da dieses einem anderen Namensraum (`http://www.tei-c.org/ns/Examples`) angehört. In der RELAX NG-Fassung ist es Teil des Schemas.

[7] Die TEI LITE existiert darüber hinaus auch in einer P4- und P5-Version.

erste Version der TEI war, die in Form einer XML-DTD formalisiert wurde, ist sie für die Annotation linguistischer Korpora aufgrund ihrer zu starken Vereinfachung nicht geeignet:

> The so-called TEI-Lite standard was the first version of the TEI with an XML DTD; however, this version in itself is not suitable for corpus encoding, as it lacks the appropriate functionality, being a drastically impoverished version of the full TEI standard. (Bański 2001, S. 3).

Eine weitere Modifikation der P3 stellt der CORPUS ENCODING STANDARD (CES) dar, der in Kapitel 7 diskutiert wird. Für die P4 existiert mit PIZZACHEF ein Web-basiertes Werkzeug zur Erstellung einer eigenständigen TEI-Anpassung, mit dessen Hilfe in sechs Schritten auf Grundlage einer Basis (Schritt 1: Wahl eines geeigneten Basismoduls), weiterer Zutaten (Schritt 2: Hinzufügen zusätzlicher Module, hier „Toppings“, also Belag, genannt) und eines Entitäten-Zeichensatzes (Schritt 3) eine „Pizza“ erstellt werden konnte, die sich in den Schritten 4 (Entfernen oder Überarbeiten von einzelnen Elementen) und 5 (Hinzufügen TEI-externer Elemente) verfeinern ließ – der sechste und letzte Schritt bestand dann im eigentlichen Pizzabacken, also der Generierung einer entsprechenden XML-DTD.[8] Mit der P5 gibt es mit ROMA einen deutlich ausgebauten Nachfolger, der alternativ auch als Download zur Verfügung steht.[9] ROMA und das für MacOS X und Linux verfügbare Desktop-Pendant VESTA[10] erlauben nicht nur noch feinere Modifikationen an der erstellten Dokumentgrammatik, sondern erstellen das resultierende Schema sowohl als RELAX NG, DTD oder XSD. Ein *Sanity Checker* überprüft vor der abschließenden Generierung die zu erstellende Grammatik auf Schwachstellen und Fehler. Hintergrund dieses neuen Anpassungswerkzeuges ist die Auszeichnungssprache ODD („One Document Does it All“), die aus einer Teilmenge der TEI besteht. Eine ODD-Instanz ist ein TEI-Dokument, das das *tagdocs*-Modul nutzt, in dem Elemente zur Definition eines neuen TEI-konformen Schemas enthalten sind – die TEI definiert sich also in einem rekursiven Szenario selbst.[11] Mittels einer solchen ODD-Instanz werden nicht nur Anpassungen in Bezug auf TEI-eigene Elemente und Attribute vorgegeben, es können auch TEI-externe Konstrukte damit eingebunden werden. Als TEI-Dokument gespeichert kann eine solche Anpassungsdatei dann als Ausgangspunkt für weitere Modifikationen dienen.

Die Module der TEI werden jeweils in einzelnen Kapiteln der TEI GUIDELINES diskutiert. Die bereits erwähnten Module *core* (Kapitel 3), *header* (Kapitel 2) und *textstructure* (Kapitel 4) befinden sich zu Beginn des Hauptteils der TEI GUIDELINES. Daran schließen

[8] Der PIZZACHEF ist weiterhin unter der Adresse `http://www.tei-c.org/pizza.html` zu finden, zuletzt abgerufen am 19.04.2012.

[9] Die Web-basierte Version befindet sich unter der Adresse `http://www.tei-c.org/Roma/`, zuletzt abgerufen am 19.04.2012.

[10] Download der MacOS X-Version unter `http://sourceforge.net/projects/tei/files/Vesta/`, zuletzt abgerufen am 19.04.2012. Für Linux-Distributionen, die das Debian-Paket-Management nutzen, stehen unter `http://tei.oucs.ox.ac.uk/teideb/` entsprechende Pakete bereit, zuletzt abgerufen am 19.04.2012.

[11] Einführungen in ODD finden sich in Bauman und Flanders (2004); Flanders und Bauman (2010); Cummings und Rahtz (2011) oder auf den Seiten des Projekts „TEI By Example“ unter der URL `http://tbe.kantl.be/TBE/modules/TBED08v00.htm`, zuletzt abgerufen am 19.04.2012.

sich die Ausführungen zum Modul *gaiji* zur Darstellung von Zeichen und Glyphen ohne Enkodierstandard (beispielsweise bei sehr alten Sprachen, Kapitel 5), zum Modul *verse* zur Annotation von Texten in Versstrukturen (Kapitel 6), zum Modul *drama* (Transkriptionen oder Skripte von Aufführungen, Kapitel 7) und das Modul *spoken* für die Annotation transkribierter Sprachdaten (Kapitel 8) an. Lexikalische Ressourcen werden mit Hilfe des Moduls *dictionaries* (Kapitel 9) annotiert, handschriftliche Aufzeichnungen oder sonstige Manuskripte mittels der Elemente und Attribute aus dem Modul *msdescription* (Kapitel 10) beschrieben, während die Repräsentation solcher Primärdaten durch Komponenten des Moduls *transcr* (Kapitel 11) erfolgt. Anmerkungen wie variierende Lesarten, orthographische Varianten und weitere Formen der Textkritik sind Gegenstand des Moduls *textcrit*, das in Kapitel 12 behandelt wird. Die nähere Beschreibung von Eigennamen, Orten und Datumsangaben kann über die Elemente und Attribute des Moduls *namesdates* erfolgen (Kapitel 13). Dieses Modul erweitert damit die allgemeineren Auszeichnungen, die bereits durch das Modul *core* bereitgestellt werden. Die Annotation von Abbildungen, Tabellen, Formeln und weiteren nicht-textuellen Bestandteilen wird in Kapitel 14 bzw. dem Modul *figures* erörert. Kapitel 15 behandelt TEI-Komponenten zur Strukturierung linguistischer Korpora (Modul *corpus*), auf die in Abschnitt 5.4 noch weiter eingegangen wird, Teile der Module *linking* (Kapitel 16) und *analyis* (Kapitel 17) sind Gegenstand des Abschnitts 5.5, während die TEI-Merkmalsstrukturbeschreibungen (Modul *iso-fs*, Kapitel 18) im Kapitel 6 dieser Arbeit separat behandelt werden. Das Modul *nets* enthält Methoden zur Repräsentation von Graphen, Bäumen und Netzwerken (Kapitel 19), *certainty* zur Kennzeichnung problematischer Stellen bzw. zur Angabe alternativer Kodierungen. Schließlich dient das bereits erwähnte Modul *tagdocs* (Kapitel 22) der Beschreibung von Auszeichnungssprachen inkl. der TEI GUIDELINES selbst.

5.3 Aufbau von TEI-Instanzen

Ein TEI-Dokument besteht aus dem Wurzelelement `TEI` und enthält zumindest ein Element `teiHeader` (für Metadaten) sowie ein `text`-Element (für den eigentlichen Inhalt). Der Namensraum des Wurzelelements ist `http://www.tei-c.org/ns/1.0`. Der `teiHeader` besteht aus insgesamt vier größeren Teilen. Zunächst erfolgt die obligatorische Angabe einer kurzen Beschreibung der vorliegenden Instanz, in Form des `fileDesc`-Elements, das wiederum aus den Elementen `titleStmt`, `publicationStmt` und `sourceDesc` besteht (vgl. Listing 5.1). Daneben bilden die optionalen Elemente `encodingDesc`, `profileDesc` und `revisionDesc` die weiteren drei wesentlichen Komponenten. Die Beziehung zwischen Originaltext und TEI-Instanz wird durch das Element `encodingDesc` (*encoding description*) beschrieben. Hier können Angaben zur Transkription (inkl. Vor- und Nachbearbeitung, beispielsweise der kanonischen Auflösung von Ambiguitäten) entweder in Form von Fließtext (unterteilt durch `p`-Elemente) oder mittels weiterer Kindelemente strukturiert gespeichert werden (vgl. P5 1.9.1, Kapitel 2, „The TEI Header“).

Das sogenannte Textprofil (*text profile*), dessen Informationen mit Hilfe des Elements `profileDesc` zusammengefasst werden, besteht aus Metadaten, die nicht bibliogra-

phischer Art sind, d. h., nicht zur Identifikation des Werkes beitragen. Dazu zählen laut Giordano (1998) Angaben über Sprecher und die Situation, in der beispielsweise Diskurse aufgenommen wurden, aber auch eine Klassifizierung des Textes (mittels des `textClass`-Elements).

Das Element `revisionDesc` kann genutzt werden, um die Änderungshistorie der Instanz zu speichern. Unterhalb von `revisionDesc` werden die einzelnen Modifikationen entweder in Form eines `list`-Elements oder durch jeweils ein `change`-Element repräsentiert. Obwohl die Aufzeichnung einer solchen Historie optional ist, wird sie sowohl für die eigentliche Annotation als auch den Header dringend empfohlen, um die Verarbeitung größerer Korpora zu erleichtern (vgl. auch Abschnitt 13.3.4).[12]

Eine weiterhin recht aktuelle Darstellung der `teiHeader`-Metadaten findet sich – neben den Ausführungen in P5 1.9.1, Kapitel 2 – in Giordano (1995); Giordano (1998), während Dunlop (1995) Überlegungen bzgl. der Metadaten für größere Korpora anstellt.

Listing 5.1: Grundaufbau einer TEI-Instanz

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <TEI xmlns="http://www.tei-c.org/ns/1.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.tei-c.org/ns/1.0 fullTEI/fullTEI.xsd">
5   <teiHeader>
6    <fileDesc>
7     <titleStmt>
8      <title>Eine minimale P5-Instanz</title>
9     </titleStmt>
10    <publicationStmt>
11     <p>Zur Demonstration.</p>
12    </publicationStmt>
13    <sourceDesc>
14     <p>Neu angelegt.</p>
15    </sourceDesc>
16   </fileDesc>
17  </teiHeader>
18  <text>
19   <body>
20    <p>Ein Absatz.</p>
21   </body>
22  </text>
23 </TEI>
```

Der Text (also die eigentlichen Daten) wird üblicherweise mit Hilfe eines `body`-Elements kodiert, optional begleitet von `front`- und `back`-Element (*Frontmatter* bzw. *Backmatter*, Vor- und Abspann – z. B. Inhaltsverzeichnis, Anhang, etc.). Unterhalb von `body` finden sowohl Einheiten der logischen Textstruktur als auch Hervorhebungen (in Form von Formatierungen oder mit Hilfe semantischer Auszeichnung) und Elemente der Textanalyse Platz.[13] Dabei finden die Elemente und Attribute ihre Entsprechung in den Kapiteln der P5 1.9.1. Relevant für die vorliegende Arbeit ist hier vor allem das Kapitel „Language Corpora“ (P5 1.9.1, Kapitel 15), aber auch die Kapitel „Linking, Segmentation,

[12] Ähnlich dazu werden in größeren Softwareprojekten ebenfalls Versionierungssysteme wie CVS, SUBVERSION oder GIT eingesetzt, um Modifikationen am Quelltext transparent nachvollziehen zu können.

[13] Bauman und Catapano (1999) diskutieren Möglichkeiten, neben dem Text auch das ihn beinhaltende physikalische Buch zu beschreiben.

and Alignment“ (P5 1.9.1, Kapitel 16), „Simple Analytic Mechanisms“ (P5 1.9.1, Kapitel 17) und „Non-hierarchical Structures“ (P5 1.9.1, Kapitel 20), sowie die Ausführungen zu Merkmalsstrukturbeschreibungen, die in Kapitel 6 gesondert diskutiert werden. Die prinzipiell für die Linguistik ebenfalls einsetzbaren Einheiten für Transkriptionen oder Wörterbücher werden hier nicht weiter behandelt.[14]

5.4 Linguistische Annotation

Die in Kapitel 15 näher erläuterten Elemente und Attribute zur Strukturierung von Korpora sollen in diesem Abschnitt kurz aufgeführt werden.[15] Zu nennen wäre hier vor allem das alternative Wurzelelement `teiCorpus`, das neben einem `teiHeader` (in diesem Fall für das gesamte Korpus) eine Anzahl von n (mit $n \geq 1$) `TEI`-Elementen mit eigenständigen Korpuseinträgen beinhaltet (die dem grundlegenden Aufbau aus Listing 5.1 folgen).

Listing 5.2: Grundgerüst einer TEI-Korpusinstanz

```
<?xml version="1.0" encoding="UTF-8"?>
<teiCorpus xmlns="http://www.tei-c.org/ns/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"
 xsi:schemaLocation="http://www.tei-c.org/ns/1.0 fullTEI/fullTEI.xsd">
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>Korpusinstanz</title>
   </titleStmt>
   <publicationStmt>
    <p><!-- [...] --></p>
   </publicationStmt>
   <sourceDesc>
    <p><!-- [...] --></p>
   </sourceDesc>
  </fileDesc>
 </teiHeader>
 <TEI>
  <teiHeader>
   <!-- [...] -->
  </teiHeader>
  <text>
   <!-- [...] -->
  </text>
 </TEI>
 <TEI>
  <!-- [...] -->
 </TEI>
</teiCorpus>
```

Die Metadaten im `teiHeader` können bei Verwendung des *corpus*-Moduls durch die Elemente `textDesc`, `particDesc` und `settingDesc` erweitert werden. Dabei beschreibt

[14] Vgl. auch die Ausführungen auf der Webseite der TEI-Arbeitsgruppe „SIG:TEI for Linguists“ unter `http://www.tei-c.org/Activities/SIG/TEI_for_Linguists/index.xml`, zuletzt abgerufen am 19.04.2012.

[15] Empfehlungen zur Annotation linguistischer Korpora mittels der TEI P3 finden sich auch in Burnard (1997).

`textDesc` die Situation, in der der Text entstand, `particDesc` die Teilnehmer des gesprochenen Textes als Ausgangspunkt einer Transkription und `settingDesc` weitere situative Merkmale (in Form von `setting`-Unterelementen, die ihrerseits Prosa oder weitere Kindelemente enthalten können).

Auch wenn mit dem Kapitel „Language Corpora“ ein vollständiges Kapitel zur Strukturierung von Sprachkorpora existiert, geben die TEI GUIDELINES in Bezug auf ein Instrumentarium zur Beschreibung linguistischer Phänomene nur wenig Vorgaben und verweisen auf die allgemeinen Möglichkeiten der Auszeichnung bis hinab auf die Zeichenebene. In der Tat ist der relevante Abschnitt 15.4, „Linguistic Annotation of Corpora“ in P5 1.9.1, S. 480f. nur eine Seite lang. Das ist nicht weiter verwunderlich, da die Spezifikation sich selbst als unabhängig von bestimmten linguistischen Theorien betrachtet (vgl. Abschnitt 5.1). Darüber hinaus empfehlen die Richtlinien, bestimmte Merkmale des Tagsets anzupassen, um die Kodierung größerer Korpora zu ermöglichen (P5 1.9.1, Abschnitt 15.5, „Recommendations for the Encoding of Large Corpora“). Auch hierzu kommen ODD-Dokumente (vgl. Abschnitt 5.2) zum Einsatz.

5.5 Multiple Annotationen

Wie bereits in Abschnitt 3.3.2 dieser Arbeit ausgeführt, stellen multiple Annotationen XML-basierte Auszeichnungssprachen vor gewisse Probleme, da diese im Fall von überlappenden Strukturen nicht in Form eines Baums repräsentiert werden können. Allerdings bietet auch die TEI eine Reihe an Ansätzen zur Umgehung dieser Problematik an, darunter neben den Merkmalsstrukturen (vgl. Kapitel 6) auch die Standoff-Annotation (vgl. Abschnitt 5.6) und die in diesem Abschnitt erläuterten Mechanismen.[16] Diese Ansätze gehen zu weiten Teilen auf die Arbeiten der Arbeitsgruppe zur Behandlung überlappender Strukturen, „SIG:Overlap“ zurück.[17] Als Grundvoraussetzung existieren in der TEI mehrere Segmentierungshierarchien: Als „chunk“ werden in den TEI GUIDELINES Absatz-bildende Elemente genannt, die als Model `model.divPart` zusammengefasst werden; typische Vertreter sind `p` (Absatz) oder `ab` (anonymer Block), aber auch die Elemente `graph` (zur Darstellung von Graphen) und `forest` (als Gruppierung mehrerer Bäume). Unterhalb der genannten Konstrukte können – je nach Kontext – die in P5 1.9.1, Abschnitt 17.1, „Linguistic Segment Categories“ aufgeführten Segmentierungseinheiten für linguistische Phänomene auftreten. Diese lassen sich kategorisieren in oberhalb und einschließlich der Wortebene, und darunter. Zur ersten Gruppe gehören die `s` (*s-unit*, ein Satz-ähnlicher Textabschnitt), `cl` (*clause*), `phr` (*phrase*, also eine Phrase im grammatischen Sinn) und `w` (*word*, ein Wort im grammatischen – nicht zwingend orthographischen – Sinn). In der zweiten Gruppe stehen die Elemente `c` (*character*, zur Auszeichnung einzelner Zeichen), `m` für Morpheme und `pc` für Interpunktion (*punctuation character*) zur Verfügung. Alle vormals genannten Elemente können mittels des `part`-Attributs zur Auszeichnung fragmentierter Strukturen herangezogen werden,

[16] Eine weitere Abhandlung über Lösungsansätze, die auf TEI-Instanzen beruhen, findet sich in Bauman (2005).

[17] Vgl. die Ausführungen auf der Homepage der SIG unter `http://wiki.tei-c.org/index.php/SIG:Overlap`, zuletzt abgerufen am 19.04.2012.

wobei die folgenden Attributwerte gültig sind: *Y* (*yes*, das Segment ist fragmentiert, also unvollständig), *N* (*no*, das Segment ist vollständig – dieser Wert wird angenommen, wenn das Attribut nicht mit einem anderen Wert gesetzt ist, Standardwert), *I* (*initial*, dieses Segment markiert den Anfang eines größeren, fragmentierten Segments), *M* (*medial*, dieses Segment ist der mittlere Teil eines größeren, fragmentierten Segments) und *F* (*final*, bezeichnet den letzten Teil eines fragmentierten Segments). Die letzten drei Werte sollten nur dann eingesetzt werden, wenn die Wiederherstellung des größeren Segments eindeutig möglich ist (z. B. durch entsprechende ID-Markierungen, vgl. P5 1.9.1, Abschnitt 17.1, „Linguistic Segment Categories“). Elemente wie das in P5 1.9.1, Abschnitt 16.3, „Blocks, Segments, and Anchors“ eingeführte `seg` repräsentieren weitere beliebige Segmentierungen unterhalb der „chunk“-Ebene. Das in Listing 5.3 dargestellte und aus P5 1.9.1, Kapitel 17 entnommene Beispiel zeigt eine solche Basissegmentierung.

Listing 5.3: Segmentierungen mittels seg-Element

```
<seg xml:id="bl0034" type="sentence">
 <seg xml:id="bl0034.1" type="phrase">Literate and illiterate speech</seg>
 <seg xml:id="bl0034.2" type="phrase">in a language like English</seg>
 <seg xml:id="bl0034.3" type="phrase">are plainly different.</seg>
</seg>
```

Anhand von eindeutigen Bezeichnern (per `xml:id`-Attribut) und Typisierungen können diese allgemeinen Segmente als Basisebene für darauf aufbauende Annotationen fungieren (z. B. mittels der ebenfalls in P5 1.9.1, Abschnitt 16.9, „Stand-off Markup“ definierten Zeiger und Verknüpfungen).

Speziell für die Darstellung überlappenden Annotationen sind in der TEI seit der P2 Milestone-Elemente spezifiziert (vgl. P2, Abschnitt 22.3.4.3, „Milestone method“). In der aktuellen Version P5 finden sich die Ausführungen dazu in Kapitel 20.[18] Wie bereits in Abschnitt 3.3.2.2 dargelegt, handelt es dabei um leere Elemente (differenzierbar in solche mit und ohne semantischen Gehalt in Form eines Elementnamens) zur Festlegung von Annotationsbeginn und -ende. Da es hierbei nicht zu Überlappungen kommen kann (vgl. auch die Ausführungen zu „Spurious Overlap“ und die Beispiele 3.6 und 3.7 auf Seite 39 in Abschnitt 3.3.2), können auf diesem Wege theoretisch beliebig viele Annotationsebenen über ein Primärdatum aufgespannt werden. Mittels Attributen werden bei den generischen Milestone-Elementen `milestone` und `anchor` die ursprünglich in Form von Elementnamen kodierten Informationen spezifiziert. Das Listing 5.4, entnommen aus P5 1.9.1, Kapitel 20, S. 624, demonstriert eine TEI-konforme Annotation der beiden Interpretationen der Phrase „Fast trains and planes“.

Durch die `anchor`-Elemente mit dem Untertyp *NPstart* wird der Beginn jeweils einer Nominalphrase gekennzeichnet, so dass sowohl die NP bestehend aus „trains and planes“ als auch die NP bestehend aus „Fast trains“ kodiert werden kann.[19] Damit lassen Milestone-Elemente eine TEI-konforme Annotation zu, die weiterhin in die Primärdaten eingebettet bleibt und damit die Lesbarkeit (bei einer nicht allzu hohen Anzahl von

[18] Das generische Element `milestone` ist allerdings Bestandteil des *core*-Moduls und wird in P5 1.9.1, Abschnitt 3.10.3, „Milestone Elements“ behandelt.

[19] Das Beispiel dokumentiert interessanterweise einen Fall von „Self Overlap“, vgl. Abschnitt 3.3.2.

Listing 5.4: Kodierung zweier Lesarten mittels TEI-Milestone-Elementen

```
<phr function="NP">
 <anchor type="delimiter" subtype="NPstart" xml:id="NPInterpretationB"/>
 <w function="A">Fast</w>
 <anchor type="delimiter" subtype="NPstart" xml:id="NPInterpretationA"/>
 <w function="N">trains</w>
 <anchor type="delimiter" subtype="NPend" corresp="#NPInterpretationB"/>
 <w function="C">and</w>
 <w function="N">planes</w>
 <anchor type="delimiter" subtype="NPend" corresp="#NPInterpretationA"/>
</phr>
```

Annotationsebenen) und die Wartung erleichtert. Wie DeRose (2004, S. 6) darlegt, können damit nicht nur zusammenhängende Einheiten annotiert werden: „Discontiguous components can be supported by adding methods such as next/prev or joins. Milestone markup conforms to XML and can be *parsed* with existing XML tools."

Allerdings ist die Möglichkeit der Verarbeitung von Annotationen, deren Start- und Endpunkt durch leere Elemente gekennzeichnet werden, beschränkt, wie DeRose (2004, S. 7) weiter ausführt:

> The main disadvantage of milestones is, however, that their XML-processability is limited. XML knows about the endpoints, but does not know that they are related or that they delimit a component that should be treated basically like elements. For example, both CSS and XSL use inheritance to apply formatting properties to the entire content of a given XML element, but they cannot readily do so for milestone-delimited ranges because they only know about true XML elements.

Darüber hinaus zeigt das Beispiel aus Listing 5.4 auch eine Schwachstelle dieser Methode: Die Zuordnung von Start- und Ende des von der Annotation umspannten Bereichs der Primärdaten ist in diesem Fall nur durch das zusätzliche Attribut `corresp` eindeutig festgelegt – üblicherweise wird davon ausgegangen, dass ein Ende-Begrenzer jeweils den ihm am nächsten stehenden Start-Begrenzer schließt, bzw. ein Milestone-Element sowohl Ende einer vorhergehenden Annotation als auch Startpunkt einer neuen sein kann (P5 1.9.1, Kapitel 20, S. 623).

Eng verwandt mit den Milestones ist der Ansatz der Fragmentierung (*Fragments* und *Virtual Joins*). Hierbei werden bestehende Elemente in kleinere Einheiten aufgeteilt, die sich überlappungsfrei in die Primärdaten einbetten lassen (*Partial Elements*, P5 1.9.1, Kapitel 20, S. 625). Zusätzlich werden diesen Teil-Vorkommen mittels des bereits erwähnten globalen `part`-Attributs Positionen innerhalb der rein virtuell vorhandenen Annotation zugewiesen (vgl. Listing 5.5 aus P5 1.9.1, Kapitel 20, S. 626).

Alternativ können die Attribute `next` und `previous` zum Einsatz kommen, die eine Kette von Element-Fragmenten auszeichnen (*Chaining*, vgl. Listing 5.6 aus P5 1.9.1, Kapitel 20, S. 626).

Das ebenfalls vorhandene Element `join` (Bestandteil des Moduls *linking* und beschrieben in P5 1.9.1, Abschnitt 16.7, „Aggregation") kann genutzt werden, um Elemente

Listing 5.5: Fragmentierung mittels part-Attribut

```
<lg>
 <l>
  <seg>Scorn not the sonnet;</seg>
  <seg part="I">critic, you have frowned,</seg>
 </l>
 <l>
  <seg part="F">Mindless of its just honours;</seg>
  <seg part="I">with this key</seg>
 </l>
 <l>
  <seg part="F">Shakespeare unlocked his heart;</seg>
  <seg part="I">the melody</seg>
 </l>
 <l>
  <seg part="F">Of this small lute gave ease to Petrarch's wound.</seg>
 </l>
</lg>
```

Listing 5.6: Fragmentierung und Chaining

```
<lg>
 <l>
  <seg>Scorn not the sonnet;</seg>
  <seg next="#s2b" xml:id="s2a">critic, you have frowned,</seg>
 </l>
 <l>
  <seg prev="#s2a" xml:id="s2b">Mindless of its just honours;</seg>
  <seg next="#s3b" xml:id="s3a">with this key</seg>
 </l>
 <l>
  <seg prev="#s3a" xml:id="s3b">Shakespeare unlocked his heart;</seg>
  <seg next="#s4b" xml:id="s4a">the melody</seg>
 </l>
 <l>
  <seg prev="#s4a" xml:id="s4b">Of this small lute gave ease to Petrarch's wound.</seg>
 </l>
</lg>
```

anhand ihres eindeutigen Bezeichners virtuell zusammenzufassen. Ein Beispiel dafür findet sich in Listing 13.16 auf Seite 257 in Abschnitt 13.3.2 dieser Arbeit.[20]

Milestones und Fragmentierungen gelten neben der strukturierten Speicherung in separaten Dokumenten als Standardlösungen zur Speicherung überlappender Annotationseinheiten und sind daher in Überblicksarbeiten zu diesem Thema wie Bauman (2005); Sperberg-McQueen (2007b); Marinelli *et al.* (2008); Stührenberg und Goecke (2008) zu finden. Dass sich Milestones auch zur Darstellung von ANNOTATION GRAPH-Strukturen nutzen lassen, zeigt Loiseau (2007), der zu diesem Zweck Milestones und hierarchische Strukturen kombiniert.

[20] Bański (2010) bewertet das Element `join` als „Local Stand-off“. Vgl. auch die Ausführungen im folgenden Abschnitt.

5.6 Standoff-Annotation

Wie bereits in Abschnitt 3.3.2.3 erläutert, stellt die Standoff-Annotation, also die Trennung von Primärdaten und Auszeichnung und anschließende Verknüpfung eine Variante der Vermeidung überlappender Strukturen dar, die auch von den TEI GUIDELINES unterstützt wird. In der P3 findet sich zwar noch nicht der Terminus „Standoff“, in P3, Abschnitt 14.9, „Connecting Analytic and Textual Markup“ wird allerdings auf entsprechende Möglichkeiten hingewiesen,

> for analytic and interpretive markup to be represented outside of textual markup, either in the same document or in a different document. The elements in these separate domains can be connected, either with the pointing attributes `ana` (for „analysis“) and `inst` (for „instance“), or by means of `<link>` and `<linkGrp>` elements.

Die angesprochenen Methoden beziehen sich auf die bereits in Abschnitt 5.5 dieser Arbeit diskutierten linguistischen Auszeichnungen sowie auf die gesondert in Kapitel 6 behandelten Merkmalsstrukturen. Da die P4 als Migration von SGML zu XML keine wesentlichen inhaltlichen Änderungen mit sich bringt, findet sich dort dieser Passus analog (P4, Abschnitt 14.9, „Connecting Analytic and Textual Markup“).

Dem gegenüber wurde in der aktuellen Version P5 der Standoff-Annotation ein eigenes Unterkapitel gewidmet (P5 1.9.1, Abschnitt 16.9, „Stand-off Markup“), die oben zitierten Aussagen finden sich allerdings weiterhin im darauf folgenden Abschnitt.[21] Neu in dieser Fassung der GUIDELINES ist die Einführung eines generischen Konzepts, um beliebige Auszeichnungen in Standoff-Notation ausdrücken zu können (nicht ganz unähnlich dem Ansatz, den das in Teil III vorgestellte XSTANDOFF nutzt). Dadurch kann jede TEI-Instanz entweder inline oder standoff repräsentiert werden (in P5 1.9.1, Kapitel 16, S. 529 als „internal markup“ bzw. „stand-off markup“ bezeichnet), um eine Instanz von einer Repräsentation in eine andere zu überführen, werden die Prozesse „internalize“ und „externalize“ definiert. Die TEI GUIDELINES setzen zur Realisierung dabei auf XML INCLUSIONS (XINCLUDE, Marsh und Orchard 2004; Marsh, Orchard und Veillard 2006), die wiederum auf das XPOINTER FRAMEWORK bzw. das XPOINTER ELEMENT() SCHEME aufbauen.

XINCLUDE definiert einen Namensraum `http://www.w3.org/2001/XInclude` und zwei Elemente `include` und `fallback` und wird dazu eingesetzt, innerhalb einer XML-Instanz Inhalte einer separaten Datei einzubetten, wobei diese Inhalte entweder Text (ohne jegliche Annotation) oder XML-annotiert sein können (deutlich gemacht durch den Wert des `parse`-Attributs des Elements `include`). Der Parser ersetzt zur Laufzeit die per `include` angelegte Referenz durch den Inhalt der über den Wert des `href`-Attributs referenzierten Datei, wobei durch das optionale Attribut `xpointer` auf ein durch einen XPOINTER spezifiziertes Fragment der Datei verwiesen werden kann. Die folgenden Beispiel-Listings aus P5 1.9.1, Kapitel 16 zeigen eine XML-Instanz mit XINCLUDE-Referenz (Listing 5.7), die einzubettende Datei (Listing 5.8) und das Resultat,

[21] Darüber hinaus gibt es in Kapitel 20 einen weiteren kurzen Abschnitt zur Standoff-Notation, der hier nur der Vollständigkeit halber erwähnt wird.

das vom Parser zur Laufzeit erzeugt wird (Listing 5.9).[22]

Listing 5.7: Datei mit XInclude

```
1 <body xmlns:xi="http://www.w3.org/2001/XInclude">
2  <div>
3   <xi:include href="ex.xml" xpointer="range(xpath1(id(p1)//emph),xpath1(id(par2)//emph))"
        parse="xml">
4    <xi:fallback>
5     <p xml:id="p_fallback">Fallback text</p>
6    </xi:fallback>
7   </xi:include>
8  </div>
9 </body>
```

Listing 5.8: Datei ex.xml

```
1 <body>
2  <p xml:id="p1">home, <emph>home</emph> on Brokeback Mountain.</p>
3  <p xml:id="p2">That was the <emph>song</emph> that I sang</p>
4 </body>
```

Listing 5.9: Resultat nach Verarbeitung der XInclude-Anweisung

```
1 <body xmlns:xi="http://www.w3.org/2001/XInclude">
2  <div>
3   <p xml:id="p1"><emph>home</emph> on Brokeback Mountain.</p>
4   <p xml:id="p2">That was the <emph>song</emph></p>
5  </div>
6 </body>
```

Prinzipiell ähnelt XInclude damit den externen allgemeinen Entitäten, die aus SGML und XML bekannt sind, allerdings gibt es eine Reihe von Unterschieden, die in Marsh, Orchard und Veillard (2006, Abschnitt 1.2, „Relationship to XML External Entities“) aufgeführt werden: Während Entitäten aufgrund ihrer Definition in der DTD erst zum Zeitpunkt des Parsings aufgelöst werden, arbeiten XInclude-Anweisungen auf Basis des XML Infosets (vgl. die Ausführungen in Abschnitt 3.3) und sind damit unabhängig von validierenden Parsern. Ein weiterer Vorteil besteht darin, dass sie nicht vorab deklariert werden müssen (Entitäten müssen benannt werden), sondern aufgrund einer URL referenziert werden können. Ist eine durch eine allgemeine externe Entität referenzierte Datei nicht erreichbar, bricht der Parser mit einem schwer wiegenden Fehler (*fatal error*) die Verarbeitung ab – dagegen bietet das `fallback`-Element als optionales Kind von `include` einen Mechanismus für den Fall, dass die per XInclude-Anweisung verwiesene Ressource nicht erreichbar ist (vgl. Zeile 5 in Listing 5.7).[23] Allerdings wird in P5 1.9.1 explizit empfohlen, das `fallback`-Element nicht zu verwenden, um das reguläre Verhalten von Standoff-Architekturen sicherzustellen.

Zu beachten ist, dass Marsh, Orchard und Veillard (2006, Abschnitt 4.2, „Included Items when `parse=,xml‘`“) für XInclude-Implementationen nur die Unterstützung des XPointer Framework und des XPointer element() Scheme vorschreiben:

[22] Die Beispiele wurden leicht abgewandelt. Sie stellen keine gültigen TEI-Instanzen dar.

[23] In diesem Fall würde ein *Resource Error* gemeldet (Marsh, Orchard und Veillard 2006, Abschnitt 4.2, „Included Items when `parse=,xml‘`“).

> The document information item of the acquired infoset serves as the inclusion target unless the xpointer attribute is present and identifies a subresource. XPointers of the forms described in [XPointer Framework] and [XPointer element() scheme] must be supported. XInclude processors optionally support other forms of XPointer such as that described in [XPointer xpointer() Scheme].

Die Unterstützung des in Listing 5.7 in Zeile 3 gezeigten XPOINTER-Ausdrucks unter Verwendung des `range`-Schemas zur Angabe eines durch Start- und Endpunkte definierten Bereichs ist also optional – was den Nutzen von XINCLUDE abhängig von der verwendeten Software macht. Gleiches gilt für das XPOINTER XPOINTER() SCHEME, das zur Referenzierung von beliebigen Bereichen genutzt werden kann und zu diesem Zweck auch in XCES – wenn auch in veralteter Form – eingesetzt wird (vgl. Kapitel 7 und insbesondere Abschnitt 7.4).

Durch XINCLUDE-Anweisungen können beliebige Teilbäume von Dateien (durch die entsprechenden Elemente) in externe Dateien ausgelagert bzw. umgekehrt auch wieder eingebunden werden – die oben genannten Prozesse „externalize“ und „internalize“. Eine Besonderheit bei diesen beiden Prozessen stellt das Vorkommen evtl. überlappender Strukturen dar. Da das Resultat des per XINCLUDE eingebundenen Dokuments bzw. Dokumentfragments wohlgeformt sein muss, werden bei Überlappungen die entsprechenden Elemente aus der einzubettenden Datei vollständig übernommen.[24] Im Beispiel-Listing 5.9 wird das deutlich: Der durch den XPOINTER-Ausdruck selektierte Bereich beginnt beim ersten `emph`-Element des Absatzes mit dem Bezeichner *p1* und reicht bis zum `emph`-Element des (zweiten) Absatzes, der durch *p2* identifiziert wird – und überlappt damit mit den Start- und Endtags der beiden `p`-Elemente. Daher werden diese beiden äußeren Elemente (bzw. das Start-Tag des ersten `p` und das End-Tag des zweiten) repliziert, um einen wohlgeformten Baum zu erzeugen. Im Umkehrschluss bedeutet das allerdings, dass XINCLUDE zur bewussten Repräsentation überlappender Annotationen nicht eingesetzt werden kann. In diesem Fall empfehlen die TEI GUIDELINES die Auslagerung der entsprechenden Hierarchien („externalize“) und deren Ersetzung durch entsprechende XINCLUDE-Verweise, die auf Teile eines minimal annotierten Primärdatendokuments verweisen.

> When overlapping hierarchies need to be represented for a single document, each hierarchy must be represented by a separate set of XInclude tags pointing to a common source document. This sort of structure corresponds to common practice in work with linguistic text corpora. In such corpora, each potentially overlapping hierarchy of elements for the text is represented as a separate stream of stand-off markup. Generally the source text contains markup for the smallest significant units of analysis in the corpus, such as words or morphemes, this information and its markup representing a layer of common information that is shared by all the various hierarchies. As a way

[24] Sofern das nicht der Fall ist, meldet die Anwendung einen schwer wiegenden Fehler: „Resources that contain non-well-formed XML result in a fatal error.“ (Marsh, Orchard und Veillard 2006, Abschnitt 4.2, „Included Items when `parse='xml'`“).

> of organizing the representation of complex data, this technique generally allows a large number of xml:id attributes to be attached to the shared elements, providing robust anchors for links and facilitating adjustments to the source document without breaking external documents that reference it. (P5 1.9.1, Kapitel 16, S. 531).

Eine minimale Annotation erscheint in diesem Fall sinnvoll, da Weißraum-Zeichen in XPOINTER-Bereichen mitgezählt werden. Durch eine grundlegende Auszeichnung können entsprechende Fehlerquellen minimiert werden, indem Zeichenpositionen nicht von Beginn des Dokuments an gezählt werden müssen, sondern von einem bestimmbaren Knoten im Baum – entweder durch die Position im Baum oder durch entsprechend eindeutige Bezeichner. Alternativ sind auch reine Textdateien (oder -knoten) als Bezugspunkte für XPOINTER-Ausdrücke nutzbar. Hier ist allerdings weiterhin zu beachten, dass in einem solchen Fall die angeführte Behandlung von Weißraum-Zeichen sowie die mangelhafte Unterstützung der entsprechenden XPOINTER-Schemata problematisch ist.

Bański (2010) stellt eine ausführliche Analyse der TEI Standoff-Auszeichnung vor und demonstriert sowohl die Möglichkeiten als auch die Versäumnisse der Spezifikation in Bezug auf diesen Mechanismus (vgl. auch Bański und Przepiórkowski 2009). Er kommt zu dem Schluss, dass trotz aller eventuellen Probleme ein großer Vorteil der TEI-Standoff-Annotation diese überwiegt: die Modularität der Auszeichnung. Darüber hinaus skizziert er – im Vergleich zu XCES (diskutiert in Kapitel 7) – als einen möglichen Lösungsansatz die Aufnahme des Elements `standoff` als Kindelement des `TEI`-Wurzelelements (z. B. durch die Aufnahme in die Model Group `model.resourceLike`), um eine Trennung von Primärdaten und Annotationsebenen weitergehend zu ermöglichen. Somit wären die in Listing 5.10 und 5.11 (in Verbindung mit Listing 5.12) gezeigten Instanzen valide.

Listing 5.10: TEI-Instanz mit lokaler Standoff-Annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI>
 <teiHeader>
  <!-- [...] -->
 </teiHeader>
 <standOff>
  <!-- [...] -->
 </standOff>
 <text>
  <!-- [...] -->
 </text>
</TEI>
```

Listing 5.11: TEI-Instanz mit externer Standoff-Annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI>
 <teiHeader>
  <!-- [...] -->
 </teiHeader>
 <text>
  <!-- [...] -->
 </text>
</TEI>
```

Listing 5.12: TEI-Instanz mit Standoff-Annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI>
 <teiHeader>
  <!-- [...] -->
 </teiHeader>
 <standOff>
  <!-- [...] -->
 </standOff>
</TEI>
```

Zusätzlich sollten entsprechende Software-Implementierungen wie LIBXML2, das bereits eine rudimentäre Unterstützung für XPOINTER bietet, personell und finanziell gefördert werden. Optional würde eine TEI-Anpassung in Form einer ODD-Instanz die entsprechenden TEI-Module für interessierte Forscher (von Bański „OWL (Ordinary Working Linguist)“ genannt) leichter zugänglich machen.

5.7 Feature Structures

Da die TEI-Merkmalsstrukturbeschreibungen im Rahmen der internationalen Normen ISO 24610-1:2006 und ISO/FDIS 24610-2 standardisiert wurden, werden diese separat in Kapitel 6 behandelt.

5.8 Verbreitung und Einsatz

Aufgrund ihres hohen Alters haben die TEI GUIDELINES eine sehr hohe Verbreitung erfahren. Beispiele dafür sind der „National Corpus of Polish“ (Narodowy Korpus Języka Polskiego, NKJP, vgl. Bański und Przepiórkowski 2010b; Bański und Przepiórkowski 2010a) und der 2004 an der Universität von Bologna erstellte „La Repubblica Corpus“, der aktuell 380 Millionen Token umfasst (Baroni *et al.* 2004).[25] Dadurch, dass der Wechsel von der ersten offiziell verfügbaren Version P3 auf die P4 vorrangig die Umstellung der Metasprache von SGML auf XML zum Ziel hatte, waren Migrationen zwischen den Versionen gut zu bewerkstelligen, weshalb viele Nutzer eine Konvertierung vornahmen. Dazu kommt, dass auch mit der aktuellen Version P5 weiterhin die P4 für eine Übergangszeit gepflegt wird und es Empfehlungen und Hilfestellungen zur Transformation zwischen diesen beiden Versionen gibt.[26]

5.9 Beurteilung

Die TEI gehört sicherlich zu den am meisten eingesetzten XML-Tagsets in der strukturierten Speicherung linguistischer bzw. geisteswissenschaftlicher Daten. In der Spezifikation liegen zwei wesentliche Meilensteine auf dem Weg hin zu austauschbaren

[25] Weitere Informationen zum NKJP finden sich unter `http://nkjp.pl/`, zum „La Repubblica Corpus“ unter `http://dev.sslmit.unibo.it/corpora/corpus.php?path=&name=Repubblica`, beide URLs zuletzt abgerufen am 19.04.2012.

[26] Informationen dazu sind unter `http://www.tei-c.org/Guidelines/P5/migrate.xml` einsehbar, zuletzt abgerufen am 19.04.2012.

Korpora begründet: die Einigung auf eine gemeinsame Syntax (in Form einer einheitlichen Metasprache, zunächst SGML, später XML) und die Verwendung eines einheitlichen Vokabulars (in Form der Elemente und Attribute, die die TEI zur Verfügung stellt). Die Festlegung einer standardisierten Metasprache als syntaktische Basis ermöglicht ein hohes Grad an Hardware- oder Software-Unabhängigkeit. Das macht die TEI relativ robust gegenüber technischen Entwicklungen. Das in den TEI GUIDELINES festgelegte einheitliche Vokabular dagegen ist anwendungsunabhängig ausgelegt, was die Verwendung in unterschiedlichen Anwendungsbereichen zulässt. Der Untersuchungsgegenstand, der Text, kann in verschiedenen Kontexten unterschiedlich aufgefasst werden: u. a. als physikalisches Objekt (von der Tontafel bis hin zum Papier), typographische Einheit (Zeichenkette, o. ä.), formales oder rhetorisches Objekt (vgl. Ide und Sperberg-McQueen 1995, 10f.) – wobei das Tagset alle genannten Varianten unterstützt, inkl. der Unterstützung von Standoff-Notation (auch wenn der Schwerpunkt auf Inline-Annotation liegt). Darüber hinaus sind die in den TEI GUIDELINES deklarierten Elemente und Attribute und deren hierarchische Strukturen Theorie-neutral ausgelegt, d. h., entweder wurde ein Konsens in der Bezeichnung entsprechender Informationseinheiten angestrebt oder es wurden alternative Auszeichnungsinventarien zugelassen.

Dennoch gibt Lavagnino (2006) einige Punkte zu bedenken, die es vorab zu klären gilt, bevor man sich für (oder gegen) die TEI GUIDELINES entscheidet. Gerade für ältere wissenschaftliche Texte gibt es Spezifikationen, die vor allem das Archivieren (und Wiederauffinden) solcher Texte durch entsprechende Merkmale erleichtern, u. a. die ENCODED ARCHIVAL DESCRIPTION (EAD).[27] Gleiches gilt für die Dokumentation von Software-Projekten: Hier steht mit DOCBOOK eine hervorragend unterstützte und zu diesem Zweck geeignete Alternative zur Verfügung. Ist die Entscheidung für die TEI gefallen, so gibt Lavagnino (2006, S. 335) zu bedenken, dass aufgrund der Komplexität ein Großteil der Elemente und Attribute als optional gekennzeichnet sind – es obliegt dem jeweiligen Autoren, die richtige Entscheidung zu treffen. Diese Problematik existiert auch bei in der TEI enthaltenen alternativen Ansätzen zur Annotation eines Merkmals, oder bei Elementen, deren Bezeichner mehrdeutig ist, und bei denen es durchaus zu Diskussionen kommen kann, welche Auszeichnung die geeignetere ist. Werden Annotatoren nicht entsprechend geschult, kann das gleiche Phänomen dann unbeabsichtigt mit unterschiedlichen Elementen ausgezeichnet werden. Hier schlägt Lavagnino analog zu den Ausführungen in (P5 1.9.1, Abschnitt 15.5, „Recommendations for the Encoding of Large Corpora“) vor, das Tagset entsprechend einzuschränken, damit solche Probleme nicht auftreten. Da die Bezeichner der Elemente und Attribute bewusst allgemein gehalten werden, kann es auch hier zu Irritationen kommen; das von Lavagnino (2006, S. 337) angeführte Beispiel des Elements `lg`, das erst über das Attribut `type` und dessen Wert *stanza* den semantischen Gehalt eines `stanza`-Elements erhält, ist offensichtlich. Eine Trennung von Konzept und konkreter Auszeichnung (Level vs. Layer) ist allerdings nicht in dem Maße gegeben, wie sie einige der anderen in diesem Teil der Arbeit diskutierten Spezifikationen ermöglichen.

Ähnlich attestiert auch Bański (2001, S. 5) der TEI eine hohe Komplexität:

> On the one hand, the TEI overgenerates in some respects, in that it offers far

[27] Weitere Informationen unter `http://www.loc.gov/ead/`, zuletzt abgerufen am 19.04.2012.

> too many ways of handling a single encoding problem. On the other, it is too poor – it offers too few options for the needs of (complex) corpora.[...] for a markup scheme to be suitable for the purpose of data interchange, it must be extremely general and flexible, so as to cover and absorb all the possible existing annotation systems. This is what the TEI is aimed to be. However, a general scheme such as the TEI allows e.g. for multiple encoding options for the same encoding problems and consequently, for a large degree of variation in encoding styles of even a single corpus.

Als Beispiel führt Bański (2001, S. 6) die weitreichenden Möglichkeiten an, Elemente ineinander zu verschachteln, so dass sich bei den drei Elementen `w`, `emph` und `orig` sechs mögliche (und valide) Varianten ergeben, ein entsprechendes Wort zu annotieren. Gerade hier setzen auf die TEI GUIDELINES aufbauende Spezifikationen wie der CORPUS ENCODING STANDARD (vgl. Kapitel 7) an, um Ambiguitäten bei der Annotation zu vermeiden. Allerdings sollte bei jedem Annotationsprojekt – unabhängig vom eingesetzten Auszeichnungsinventar – eine gründliche Schulung der Annotatoren erfolgen, die Erfahrungen aus dem „Sekimo“-Projekt zeigen, dass sich diese auszahlt (vgl. die sehr guten Werte für κ in Goecke, Stührenberg *et al.* 2008).

Das TEI-Tagset (sofern überhaupt noch von einem solchen einheitlichen gesprochen werden kann) ist seit der Version 5 mittels RELAX NG spezifiziert. Damit steht prinzipiell die volle Ausdrucksstärke einer Regulären Baumgrammatik zur Verfügung. Diese wird allerdings nicht ausgenutzt, da das RELAX-NG-Schema selbst automatisiert aus einer ODD-Datei (und damit aus einer TEI-Instanz) generiert wird, wie bereits in Bauman (2005) dargelegt wird:

> The TEI Guidelines are encoded in a single XML document which is itself a TEI document. Various forms of output may be generated from this source file, including formal reference documentation, the descriptive chapters of the TEI Guidelines, and the schema files (in various languages). In the conventional terminology of literate programming it is a „tangle“ process that produces the P5 RelaxNG schema files. Because the TEI P5 RelaxNG schema files are machine-generated, the syntax used is only a subset of that permitted by RelaxNG, and the format is 100% consistent and predictable. [...] Furthermore, because of the consistent indirection of the TEI P5 RelaxNG compact syntax schema files, the format of every element declaration is exactly the same, differing only in the name of the element being declared. An example here will be worth a thousand words. (Bauman 2005, S. 5).

Da die Dokumentgrammatik auch in Form einer XML-DTD bzw. eines XML-Schemas verfügbar ist, kann davon ausgegangen werden, dass die formale Ausdrucksstärke der einer LTG entspricht, da zumindest keine Wildcards verwendet werden, was gegen eine RCG spricht.[28]

[28] Eine umfassende Analyse auf konkurrierende Nichtteminale wurde aufgrund des Umfangs nicht durchgeführt. Für eingeebnete Dokumentgrammatiken (also solchen, die aus nur einer Datei bestehen), ist eine oberflächliche Überprüfung eines XML-Schemas mittels folgendem XPath-Ausdruck denkbar:

Aufgrund der nicht zu widerlegenden Vorteile ist die TEI in der aktuellen Version P5 allerdings ein hervorragender Kandidat zur Erstellung größerer Textkorpora.

`//xs:element[@name][name(..)='xs:choice' or name(..)='xs:sequence']`. Die von diesem Pfadausdruck zurückgelieferten Elementdeklarationen können anschließend auf einen gleichen Namen hin überprüft werden, was auf konkurrierende Nichtterminale in verschiedenen Kontexten hinweisen würde.

6

Standardisierte Beschreibung von Merkmalsstrukturen

Feature Structures (Merkmalsstrukturen, vgl. Carpenter 1992; Copestake 2002) sind in der Linguistik weit verbreitet, unter anderem werden sie in der *Head-driven Phrase Structure Grammar* (HPSG, Pollard und Sag 1987; Pollard und Sag 1994) oder bei der *Lexical Functional Grammar* (LFG, Kaplan und Bresnan 1982; Dalrymple 2001) in der F-Struktur zur Darstellung syntaktischer Kategorien eingesetzt (für die Einführung von Merkmalsstrukturen in HPSG vgl. beispielsweise Balari Ravera 1991). In diesem Sinne beschreiben Merkmalsstrukturen allgemein die Eigenschaften eines Objekts in einer abstrakten hierarchisch organisierten Form und haben den Vorteil, dass sie partielle Eigenschaften beschreiben können. Werden zwei Merkmalsstrukturen kombiniert (*unifiziert*) erhält man eine neue Merkmalsstruktur, die alle Informationen beinhaltet, die in den beiden ursprünglichen kodiert waren (und ggf. zusätzliche). Aufgrund dieser Möglichkeit lassen sich Feature Structures auch zur Abbildung multipler Annotationen einsetzen, wie Stegmann und Witt (2009) zeigen (vgl. auch Abschnitt 6.3). Grundlegender Bestandteil sind Attribut-Wert-Paare, die jeweils eine Eigenschaft (z. B. Genus, Kasus, etc.) und deren Wert (z. B. Maskulin, Akkusativ) speichern, wobei letztere ebenfalls wieder aus Merkmalsstrukturen bestehen können, so dass Feature Structures rekursiv sein können. Klassischerweise werden Merkmalsstrukturen in Form von Attribut-Wert-Matrizen (*Attribute Value Matrix*, AVM) dargestellt.

Formal gesehen sind Merkmalsstrukturen gerichtete, azyklische Graphen (mit einer Wurzel, DAG, vgl. Abschnitt 3.3.3), wobei die Knoten den Werten der Merkmale und die Kanten den Merkmalsnamen entsprechen. Eine konkrete Merkmalsstruktur ist zu unterscheiden von der Spezifikation ihres Aufbaus (die notwendig ist, um Werte und Merkmale korrekt zueinander zuzuordnen) und einer Festlegung der Merkmale und deren zulässigen Werte. Im konkreten Fall wird Erstere in XML-Notation durch ein Tagset bzw. eine Dokumentgrammatik festgelegt, Letztere durch eine *Feature System Declaration* (FSD, Merkmalssystembeschreibung), die gesondert in Abschnitt 6.2

person	
VORNAME	vorname
NACHNAME	nachname
GESCHLECHT	geschlecht

Abbildung 6.1: Einfache AVM

person	
VORNAME	max
NACHNAME	meier
GESCHLECHT	männlich

Abbildung 6.2: Einfache AVM (konkrete Instanziierung)

diskutiert wird.

Eine sehr einfache AVM ist in Abbildung 6.1 dargestellt. Das Beispiel (eine Vereinfachung der in Müller 2008, S. 30ff., aufgeführten AVM) zeigt eine typisierte Merkmalsstruktur. Am Beispiel des Objekttyps „person“ werden die Merkmale VORNAME, NACHNAME und GESCHLECHT aufgeführt, die jeweils atomare Werte annehmen können. Abbildung 6.2 zeigt eine AVM zur Beschreibung einer konkreten Person.

6.1 TEI Feature Structures

Mit der Erarbeitung eines einheitlichen Tagsets für Merkmalsstrukturen durch die TEI (P5 1.9.1, Kapitel 18, „Feature Structures“), und dessen Verabschiedung als internationale Norm ISO 24610-1:2006, steht eine ausgereifte Auszeichnungssprache für Feature Structures zur Verfügung.[1] Aufgrund der aufwendigen Erstellung von Merkmalsstrukturen kann davon ausgegangen werden, dass eine standardisierte Beschreibung (und damit die erleichterte Wiederverwendung) sowohl von Merkmalsstrukturen als auch von FSD auf allgemein große Zustimmung stoßen wird.

Die Arbeiten an den TEI Feature Structures gehen bis zur Version P1 zurück. Bereits hier finden sich Ausführungen zur Kodierung von Merkmalsstrukturen, die jeweils aus einer Anzahl von Merkmal-Wert-Paaren bestehen. Alternativ war auch die Angabe eines einzelnen atomaren Wertes möglich um in dieser frühen Fassung die (konkreten) Werte zu kodieren (vgl. Listing 6.1, das die AVM aus Abbildung 6.2 in der SGML-Notation der P1 zeigt).[2] Weitere Ausführungen zu den Gründen für die spätere Ausgestaltung der

[1] Die Überarbeitung der ISO-Norm steht seit Juli 2010 vor der Registrierung als DIS (ISO/CD 24610-1), d. h., die Phase des Committee Draft ist abgeschlossen. Gründe für die Überarbeitung dürften in der Kompatibilität mit ISO/FDIS 24610-2 (vgl. Abschnitt 6.2) zu suchen sein, ebenso ist eine Anpassung an die aktuelle Fassung der TEI GUIDELINES denkbar. Mit ISO/DIS 24610-1 steht eine Version der Spezifikation im Status eines DIS zur freien Verfügung.

[2] Die Instanz ist strukturiert anhand der dem Verfasser vorliegenden SGML-DTD `TEILING1.DTD`. Die fragmentarischen Ausführungen zur Thematik befinden sich in der Datei P182.SCR (P1, Section 8.2, im Format Waterloo Script), die dem Verfasser ebenfalls vorliegt.

Merkmalsstrukturen in der TEI P3 finden sich in Langendoen und Simons (1995).

Listing 6.1: SGML-Kodierung von Merkmalsstrukturen in der TEI P1

```
<f.struct>
 <f.struct.name>person</f.struct.name>
 <feature>
  <f.name>vorname</f.name>
  <f.struct>max</f.struct>
 </feature>
 <feature>
  <f.name>nachname</f.name>
  <f.struct>meier</f.struct>
 </feature>
 <feature>
  <f.name>geschlecht</f.name>
  <f.struct>männlich</f.struct>
</f.struct>
```

Während der Arbeiten zur P5 wurde gemeinsam von Mitgliedern des TEI-Consortiums und ISO/TC 37/SC 4 ein Vorschlag zur standardisierten Beschreibung von Merkmalsstrukturen erarbeitet. Ziel war die Vereinfachung der bestehenden Konzeption durch Fokussierung auf linguistische Aspekte (K. Lee *et al.* 2004, S. 374). Dessen Notation unterscheidet sich in einigen Punkten von der in Listing 6.1 sichtbaren: Das `fs`-Element bildet das Wurzelelement einer Feature Structure und erlaubt über ein optionales `type`-Attribut deren Typisierung (der Typ entspricht der Angabe oben links in der AVM-Notation).[3] Unterhalb von `fs` sind die Merkmale (`f`) angeordnet, die jeweils über das obligatorische Attribut `name` bezeichnet werden. Die Werte eines Merkmals werden über verschiedene Kindelemente gespeichert, die je nach Wertausprägung einen anderen Namen haben. Einfache (atomare) Werte können aus Binärwerten (`binary`), numerischen (`numeric`) oder symbolischen (`symbol`, d. h., als Wert einer Auswahlliste) Angaben bestehen.[4] Alternativ sind auch Zeichenketten (`string`) oder Kombinationen der genannten Ausprägungen erlaubt (über das `vColl`-Element). Komplexe Werte bestehen ihrerseits aus Merkmalsstrukturen (also `fs`-Elementen).

Listing 6.2 veranschaulicht die P5-Annotation des Beispiels aus Abbildung 6.2 auf der vorherigen Seite und Listing 6.1.[5] Binärwerte kommen zum Einsatz, wenn der konkrete Wert aus genau zwei ausgewählt werden kann (beispielsweise die Unterscheidung von transitiven zu intransitiven Verben). Das in Listing 6.2 genutzte Element

[3] Zu beachten ist, dass das Wurzelelement einer TEI-Instanz, die eine Merkmalsstruktur beinhaltet, weiterhin das `TEI`-Element ist. Unter der Voraussetzung, dass ein Element `p` oder `div` auf gleicher Ebene vorhanden ist, kann das Element `fs` auch unterhalb des `body`-Elements eingebettet werden. Es ist unklar, ob diese Einschränkung absichtlich ist, sie ist zumindest nicht zielgerichtet.

[4] Zur Überprüfung ob der Wert einer konkreten Merkmalsstruktur dieser Liste entstammt ist, ist eine sogenannte *Feature Library* bzw. *Feature-Value Library* notwendig. Alternativ kann auch eine *Feature System Declaration* genutzt werden (vgl. Abschnitt 6.2).

[5] Die Ausführungen in diesem Kapitel gelten sowohl für die in der TEI P5 als auch für die in den internationalen Normen ISO 24610-1:2006 und ISO 24610-2:2011 standardisierten Merkmalsstrukturen. Bei den Beispiel-Listings ist darauf zu achten, dass das Wurzelelement einer nach diesen Spezifikation gültigen XML-Instanz in den beiden letzteren Fällen entweder `fs`, `fLib` oder `fvLib` ist, wobei für die nicht-normative Dokumentgrammatik nur die in der frei verfügbaren ISO/DIS 24610-1 enthaltene herangezogen werden konnte.

`string` verdeutlicht, dass die Ausprägung der Merkmale VORNAME und NACHNAME aus einem sehr großen, evtl. unendlich großen Bereich stammen kann, hier die konkreten Zeichenketten. Im Gegensatz dazu steht die Verwendung von `symbol`, die anzeigt, dass für die Ausprägung des Merkmals GESCHLECHT nur aus einer begrenzten Anzahl von Werten ausgewählt werden kann.[6] Zu beachten ist, dass numerische Angaben (`numeric`) sowohl einzelne Zahlen als auch Zahlenbereiche abdecken können (P5 1.9.1, Kapitel 18, S. 555). Im letzteren Fall sollte über das `max`-Attribut die obere Grenze des Bereichs gesetzt werden.

Listing 6.2: Merkmalsstrukturen in der TEI P5

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.tei-c.org/ns/1.0 TEI_FS/TEI_FS.xsd"
 xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>
  <!-- [...] -->
 </teiHeader>
 <text>
  <body>
   <p></p>
   <fs type="person">
    <f name="vorname">
     <string>max</string>
    </f>
    <f name="nachname">
     <string>meier</string>
    </f>
    <f name="geschlecht">
     <symbol value="männlich"/>
    </f>
   </fs>
  </body>
 </text>
</TEI>
```

Die in Listing 6.2 kodierten Informationen lassen sich auch in einer klassischen XML-Inline-Annotation darstellen (vgl. Listing 6.3).

Listing 6.3: Alternative Darstellung der in Listing 6.2 kodierten Informationen

```
<?xml version="1.0" encoding="UTF-8"?>
<person geschlecht="männlich">
 <vorname>max</vorname>
 <nachname>meier</nachname>
</person>
```

Diese alternative Darstellung sollte bereits zwei Aspekte der allgemeinen Merkmalsstrukturen, wie sie in P5 1.9.1, Kapitel 18 und in der Norm ISO 24610-1:2006 spezifiziert sind, deutlich machen: (1) Feature Structures sind aufgrund ihres allgemeinen Aufbaus (sowohl in Bezug auf die Benennung der Elemente und Attribute als auch durch die

[6] Alternativ wäre eine Repräsentation mittels `binary` denkbar gewesen, diese hätte allerdings ein Merkmal MÄNNLICH bzw. WEIBLICH vorausgesetzt – und würde weitere Ausprägungen nicht zugelassen.

Möglichkeit der rekursiven Verschachtelung) äußerst flexibel nutzbar, (2) der Aufwand, eine Merkmalsstruktur nach dem TEI-Tagset zu erstellen, ist erheblich.

Dieser Aufwand kann dadurch gemindert werden, indem häufig verwendete Merkmal-Wert-Paare in Form von Merkmalsbibliotheken (*Feature Libraries*) gespeichert werden, auf die dann in der Beschreibung des konkreten Merkmals verwiesen wird. Sollen nur Merkmalswerte zur mehrfachen Verwendung gespeichert werden, kann zu diesem Zweck eine *Feature-Value Library* eingesetzt werden. Beide Varianten werden von den TEI GUIDELINES unterstützt. Geht man davon aus, dass das Objekt vom Typ PERSON, „Max Meier“, einen Vater hat, der den gleichen Nachnamen trägt und ebenfalls männlichen Geschlechts ist, dann können diese beiden Merkmal-Wert-Kombinationen in eine Feature Library ausgelagert werden (Element `fLib` in Listing 6.4). Nachdem beide Merkmale in der Merkmalsbibliothek angelegt wurden, kann ein komplexes Merkmal wie „männliches Familienmitglied der Familie Meier“ als Merkmalsstruktur in einer Feature-Value Library (Element `fvLib` in Listing 6.4) gespeichert und über seinen Bezeichner (im Beispiel: *ng*) referenziert werden. Dabei referenziert wiederum das komplexe Merkmal über das `feats`-Attribut die beiden atomaren Merkmale mit den Bezeichnern *n* und *g* (Zeile 11 in Listing 6.4).

Listing 6.4: Exemplarische Merkmalsbibliothek

```
1  <!-- [...] -->
2  <fLib n="familiäre merkmale">
3   <f xml:id="n" name="nachname">
4    <string>meier</string>
5   </f>
6   <f xml:id="g" name="geschlecht">
7    <symbol value="männlich"/>
8   </f>
9  </fLib>
10 <fvLib xml:id="ngLib" n="männliches familienmitglied">
11  <fs xml:id="ng" feats="#n #g"/>
12 </fvLib>
13 <fs type="person">
14  <f name="vorname">
15   <string>max</string>
16  </f>
17  <f name="meier-männlich" fVal="#ng"/>
18 </fs>
19 <!-- [...] -->
```

Auf diese Weise sinkt nicht nur der Aufwand zur Erstellung und Verwaltung komplexer Merkmalsstrukturen, auch die Möglichkeit diese mehrfach zu verwenden wird erheblich vereinfacht und mögliche Fehlerquellen werden reduziert.

Eine weitere Besonderheit von Merkmalsstrukturen ist die Möglichkeit, Teilstrukturen gemeinsam zu nutzen (Strukturteilung oder Identität bzw. *Structure Sharing* oder *Re-entrant*). Im Gegensatz zur Auslagerung von Merkmalen bzw. Merkmal-Wert-Paaren in einer Bibliothek und anschließender Referenzierung, die formal einer Kopie der entsprechenden Struktur an der Stelle der Referenz entspricht, werden in diesem Fall nur Referenzen genutzt, es handelt sich also um eine „strikte Identität“, da es um dieselben (nicht nur um die gleichen) Werte bzw. Merkmale geht.

In AVMs werden Strukturteilungen durch eine Ziffer (üblicherweise in einem Kasten) dargestellt. Abbildung 6.3 zeigt ein Beispiel. Die Werte der ersten beiden Merkmale sind atomar, die Wert der Merkmale VATER, MUTTER und KINDER dagegen sind komplex (im letzten Fall sogar in Form einer Liste von Objekttypen, die durch spitze Klammern repräsentiert wird). Darüber hinaus ist diese Merkmalsstruktur rekursiv: Als eines der Kinder der Eltern von „Max Meier“ wird auf ihn selbst verwiesen. Da sowohl Vater als auch Mutter Eltern desselben Kindes sind, werden die Kinder durch die Indizes 1 und 2 ausgezeichnet. Während die Objekte beim Vater mit ihren Merkmalen aufgeführt sind, werden sie in der Merkmalsstruktur der Mutter referenziert.

$$
\left[\begin{array}{ll}
\textit{person} & \\
\text{VORNAME} & \text{max} \\
\text{NACHNAME} & \text{meier} \\
\text{GESCHLECHT} & \text{männlich} \\
\text{VATER} & \left[\begin{array}{ll}
\textit{person} & \\
\text{VORNAME} & \text{peter} \\
\text{NACHNAME} & \text{meier} \\
\text{GESCHLECHT} & \text{männlich} \\
\text{KINDER} & \left\langle \boxed{1} \left[\begin{array}{ll} \textit{person} & \\ \text{VORNAME} & \text{klaus} \end{array}\right], \boxed{2} \right\rangle
\end{array}\right] \\
\text{MUTTER} & \left[\begin{array}{ll}
\textit{person} & \\
\text{VORNAME} & \text{anna} \\
\text{NACHNAME} & \text{meier} \\
\text{GESCHLECHT} & \text{weiblich} \\
\text{KINDER} & \left\langle \boxed{1}, \boxed{2} \right\rangle
\end{array}\right]
\end{array}\right] \quad \boxed{2}
$$

Abbildung 6.3: Attribut-Wert-Matrize mit komplexen Werten, vererbten Informationen und zyklischen Strukturen

Eine solche Strukturteilung wird in den TEI Feature Structures wie im Listing 6.5 deutlich gemacht. Dazu wird die Struktur mit einem Element `vLabel` (*Value Label*) umgeben (in Zeile 25), mit dem gleichen Element (dann allerdings ohne Inhalt) wird anschließend darauf verwiesen (Zeile 50). Im Listing fehlt die Referenz auf die Gesamtmerkmalsstruktur (entsprechend der Referenz 2 bzw. *kind2*), da in der TEI es nicht möglich ist, das Element `vLabel` direkt als Kind von `body` einzusetzen. Somit fehlt hier die Möglichkeit, eine vollständige Merkmalsstruktur auf quasi-Wurzelebene zyklisch zu referenzieren. Interessanterweise ist in P5 1.9.1, Kapitel 18 das Element `vLabel` als Kind des `f`-Elements dargestellt, was ja auch der Bezeichnung „Value Label“ entspricht, da es so als Referenz für den Wert, nicht aber für die vollständige Merkmalsstruktur dient. Unklar bleibt jedoch, ob diese Einschränkung beabsichtigt ist. Da komplexe Werte wiederum aus Feature Structures bestehen können, ist es allerdings unproblematisch,

wie in Zeile 25 ein `fs`-Element als Kind von `vLabel` zu nutzen.

Anzumerken ist noch, dass das Attribut `name` des `vLabel`-Elements den Datentyp `xs:NAME` hat, d. h., im XML-Sinne erfolgt keine echte Referenzierung über die dafür vorhandenen Integritätsmechanismen wie `ID/IDREF/IDREFS` und eine eindeutige Vergabe von Attributwerten wird nicht überprüft.[7] Als Linking-Mechanismus wird hierzu XPOINTER eingesetzt, ein prinzipiell universelles Framework, dass in der P5 die in der P4 noch vorhandenen, TEI-eigenen Extended Pointer ersetzt. Während die Verknüpfung von Knoten über `ID/IDREF/IDREFS` nur innerhalb ein und derselben XML-Instanz verfügbar ist, sind per XPOINTER realisierte Verweise auch über Dateigrenzen hinweg nutzbar. Allerdings ist hierzu die Unterstützung durch entsprechende Software notwendig.

Ebenfalls im Beispiel-Listing 6.5 sichtbar ist die Abbildung von Werte-Sammlungen mittels `vColl`-Elementen (*Collection of Values*). Solche Sammlungen sind alternative komplexe Werte, die nicht durch vollständige Merkmalsstrukturen abgebildet werden. Über das `org`-Attribut wird die Art der Sammlung festgelegt: Fehlt das Attribut, ist die Sammlung eine Liste, d. h., die Elemente darin sind geordnet und nicht zwingend eindeutig. Der Wert *set* steht für eine Menge, in der die Elemente nicht geordnet, aber eindeutig sind, ein Wert *bag* markiert sowohl das Fehlen einer Ordnung als auch das Fehlen der Eindeutigkeit der Mitglieder (P5 1.9.1, Kapitel 18, S. 563). Im Beispiel wurde der Wert *set* gewählt, da eine Reihung der Kinder nicht relevant sein sollte, diese aber eindeutig sind.

Zusätzlich zu den bisher aufgezählten Möglichkeiten, erlauben die TEI-Merkmalsstrukturen Konstruktionen, um die Werte eines Merkmals dynamisch in Abhängigkeit von anderen Merkmal-Wert-Kombinationen festzulegen. Dazu stehen mit `vAlt`, `vNot` und `vMerge` drei Elemente zur Auswahl, die an Stelle eines Wertes eingesetzt werden können. Ersteres dient als Klammer für zwei oder mehrere Werte, von denen einer (exklusives Oder) ausgewählt werden kann. Das `vAlt`-Element entspricht damit den Elementen `xs:choice` aus XML SCHEMA bzw. `choice` in RELAX NG. `vNot` dient der Negation eines Wertes (korrekter: es liefert den Komplementärwert zurück), z. B., um auszudrücken, dass der Kasus eines Nomens nicht Genitiv sein soll (P5 1.9.1, Kapitel 18, S. 568), was sich so kürzer ausdrücken lässt, als über eine Kombination aus `vAlt`-Element und drei `symbol`-Elementen. Das Elemente `vMerge` dient ähnlich wie `vColl` dazu, mehrere Werte (atomar oder komplex) zusammenzufassen, allerdings im Sinne einer Vereinigung, nicht einer Auswahlliste.

[7] Elemente oder Attribute vom Datentyp `xs:ID` folgen prinzipiell den selben Regeln wie solche vom Datentyp `xs:NCName`, mit der Einschränkung, dass deren Werte einmalig in der Instanz sein müssen. Technisch gesehen ist der Datentyp `xs:ID` eine Ableitung durch Restriktion vom allgemeineren Datentyp `xs:NCName` (van der Vlist 2003b, S. 425ff.).

Listing 6.5: Identität in Merkmalsstrukturen

```
<!-- [...] -->
<fs type="person">
 <f name="vorname">
  <string>max</string>
 </f>
 <f name="nachname">
  <string>meier</string>
 </f>
 <f name="geschlecht">
  <symbol value="männlich"/>
 </f>
 <f name="vater">
  <fs type="person">
   <f name="vorname">
    <string>peter</string>
   </f>
   <f name="nachname">
    <string>meier</string>
   </f>
   <f name="geschlecht">
    <symbol value="männlich"/>
   </f>
   <f name="kinder">
    <vColl org="set">
     <vLabel name="kind1">
      <fs type="person">
       <f name="vorname">
        <string>klaus</string>
       </f>
      </fs>
     </vLabel>
     <vLabel name="kind2"/>
    </vColl>
   </f>
  </fs>
 </f>
 <f name="mutter">
  <fs type="person">
   <f name="vorname">
    <string>anna</string>
   </f>
   <f name="nachname">
    <string>meier</string>
   </f>
   <f name="geschlecht">
    <symbol value="weiblich"/>
   </f>
   <f name="kinder">
    <vColl org="set">
     <vLabel name="kind1"/>
     <vLabel name="kind2"/>
    </vColl>
   </f>
  </fs>
 </f>
</fs>
<!-- [...] -->
```

6.2 Merkmalssystembeschreibungen

Die in P5 1.9.1, Abschnitt 18.11, „Feature System Declaration“ aufgeführten Merkmalssystembeschreibungen (*Feature System Declaration*) wurden im September 2011 als Internationaler Standard ISO 24610-2:2011 veröffentlicht. Die dem Verfasser vorliegende Version ISO/DIS 24610-2 hat den Stand eines DIS und wird zusammen mit den Ausführungen aus P5 1.9.1, Kapitel 18 als Grundlage für diesen Abschnitt herangezogen. Beide Ausführungen unterscheiden sich nur gering, allerdings ergänzt ISO/DIS 24610-2 die Ausführungen der TEI GUIDELINES um einige formale Definitionen und ist dadurch ausführlicher.

Eine FSD erfüllt nach ISO/DIS 24610-2, S. 21f. folgende Aufgaben:

- Auflistung aller Merkmale und möglicher Werte eines Merkmalssystems (inkl. optionaler ausführlicher Beschreibung).
- Möglichkeit der Überprüfung einer Merkmalsstruktur auf Validität. Das Konzept der Validität einer (getypten) Merkmalsstruktur unterscheidet sich von dem, das in XML genutzt wird. Bei der Repräsentation von FS in XML sind daher beide zu beachten: „A feature structure representation in XML is valid if and only if it is well-formed and also conforms to the feature system declared (in a DTD, an XML Schema or some other format) for a particular application that uses typed feature structures.“ (ISO/DIS 24610-2, S. 13).

  Eine getypte Merkmalsstruktur ist darüber hinaus als gültig anzusehen, wenn folgende Voraussetzungen erfüllt sind: „Then the values of its features must fall within the admissible value ranges declared. Finally, it must satisfy the type constraints imposed on it as well as the constraints inherited from its base types“ (ISO/DIS 24610-2, S. 13).

  Etwas vereinfacht ausgedrückt, bedeutet das, dass XML-Repräsentationen getypter FS neben den grammatikalischen Constraints (festgelegt durch eine Dokumentgrammatik) zusätzlichen Einschränkungen (z. B. Co-Constraints, vgl. Abschnitt 3.4.4.4) unterworfen sind.
- Vorhalten von Vorgabewerten für unterspezifizierte Merkmalsstrukturen.

Listing 6.6 zeigt den Aufbau einer exemplarischen FSD.

Listing 6.6: Aufbau einer FSD

```
<fsDecl type="POS">
 <fsDescr>Beschreibung der Merkmalsstruktur</fsDescr>
 <fDecl name="Merkmal1">
  <!-- Deklaration des Merkmals -->
 </fDecl>
 <fDecl name="Merkmal2">
  <!-- Deklaration des Merkmals -->
 </fDecl>
 <fsConstraints>
  <!-- Zusätzliche Einschränkungen der Merkmalsstruktur -->
 </fsConstraints>
</fsDecl>
```

Das in den TEI GUIDELINES und das in ISO/DIS 24610-2 dargestellte Repräsentationsformat nutzen den gleichen Aufbau, allerdings kann das Element `fsDecl`, das die Wurzel einer Merkmalsstrukturbeschreibung darstellt, je nach Spezifikation an unterschiedlichen Stellen in der Instanz stehen. In den TEI GUIDELINES kann es innerhalb des Headers eines TEI-konformen Dokuments stehen (vgl. Listing 6.7).

Listing 6.7: FSD in TEI-Notation

```
<TEI>
 <teiHeader>
  <fileDesc>
   <!-- [...] -->
  </fileDesc>
  <encodingDesc>
   <!-- [...] -->
   <fsdDecl type="POS">
    <!-- [...] -->
   </fsdDecl>
   <!-- [...] -->
  </encodingDesc>
 </teiHeader>
 <text>
  <!-- [...] -->
 </text>
</TEI>
```

Diese Variante ist auch nach ISO/DIS 24610-2 möglich (und vor allem dann sinnvoll, wenn Merkmalssystembeschreibungen und Merkmalsstrukturen in der gleichen Instanz vorkommen). Zusätzlich erlaubt die Norm die Verortung von `fsDecl` als Geschwisterknoten von `teiHeader` und führt ein neues Wurzelelement `fsd` zur Trennung von Merkmalssystembeschreibung und Merkmalsstrukturen ein (vgl. Listing 6.8).[8]

Listing 6.8: FSD in ISO-Notation

```
<fsd>
 <teiHeader>
  <!-- Metadaten -->
 </teiHeader>
 <fsDecl type="POS">
  <!-- Merkmalsstrukturbeschreibung -->
 </fsDecl>
</fsd>
```

Die Verknüpfung von FSD und Merkmalsstrukturen erfolgt entweder implizit über den Wert des `type`-Attributs (wenn beide im gleichen Dokument vorhanden sind) oder explizit über ein `fsdLink`-Element (genauer: über dessen `target`-Attribut vom Datentyp `xs:anyURI`).

6.2.1 Aufbau einer FSD

Wie bereits aus Listings 6.8 ersichtlich, besteht eine Merkmalssystembeschreibung aus dem Element `fsd`. Dieses Element ist nur Bestandteil der in ISO/DIS 24610-2, S. 24

[8] Obwohl das Element `fsd` in Beispielen in ISO/DIS 24610-2 aufgeführt ist, fehlt es in der – informativen – RELAX NG-Dokumentgrammatik. Hier wird als Start- bzw. Wurzelelement das Element `fsdDecl` deklariert, das aus mindestens einem `fsDecl` oder `fsdLink` besteht.

enthaltenen Deklaration. Eine FSD enthält eine Anzahl von Merkmalsstrukturbeschreibungen (`fsDecl` – und in der ISO-Version einen optionalen TEI-Header für Metadaten). Diese wiederum bestehen aus einer optionalen Beschreibung (`fsDescr`), mindestens einer Merkmalsdeklaration (`fDecl`) und optionalen Einschränkungen (`fsConstraints`).

Jedes Merkmal wird durch ein `fDecl`-Element deklariert. Dabei legt der Wert des `name`-Attributs den Bezeichner fest, der in der Instanziierung durch das `f`-Element (genauer: dessen `name`-Attribut) referenziert wird. Das Attribut `optional` (vom Datentyp `xs:boolean`, Vorgabewert ist *true*) zeigt an, ob der Wert dieses Merkmals vorhanden sein muss oder nicht. Ähnlich wie das Element `fsDecl` kann auch die Deklaration eines Merkmals mittels einer umgangssprachlichen Beschreibung (hier `fDescr`) näher erläutert werden. Weitere Kindelemente des `fDecl`-Elements sind `vRange` (obligatorisch) und `vDefault` (optional). Das `vRange`-Element definiert einen Bereich gültiger Werte für das Merkmal. Dazu werden bei komplexen Merkmalsstrukturen die bereits diskutierten Kindelemente `fs`, `vColl`, `vNot` oder `vMerge` genutzt, bei einfachen die Elemente `binary`, `symbol`, `numeric`, `string`, `vLabel`, `default` oder `vAlt`. Kommt das `vDefault`-Element zur Deklaration von Standardwerten zum Einsatz (diese Werte werden dann angenommen, wenn in der Instanziierung das Element `f` mit dem entsprechenden Namen nicht vorhanden ist), ist es wie folgt aufgebaut: Sofern die Vorgabewerte an keine weiteren Bedingungen gebunden sind, werden diese durch `fs`-Vorkommen oder alternativ atomare Werte realisiert. Sollen bestimmte Bedingungen überprüft werden, werden diese durch entsprechende `if`-Elemente notiert. Ein entsprechendes Beispiel für eine solche bedingte Wertzuweisung zeigt das Listing 6.9 aus P5 1.9.1, Kapitel 18, S. 580, das die Deklaration für das Merkmal „COMP“ zur Darstellung der Oberflächenform einer subordinierende Konjunktion (*Complementizer*) demonstriert.[9] Die Bedingung sagt aus, dass für eine Phrase mit einer infiniten Verbform und einem Subjekt eine solche genutzt werden muss.[10]

Listing 6.9: Beispiel für bedingte Vorgabewerte

```
<fDecl name="COMP">
 <fDescr>surface form of the complementizer</fDescr>
 <vRange>
  <vAlt>
   <symbol value="for"/>
   <symbol value="that"/>
   <symbol value="whether"/>
   <symbol value="if"/>
   <symbol value="NIL"/>
  </vAlt>
 </vRange>
 <vDefault>
  <if>
   <fs>
    <f name="VFORM">
     <symbol value="INF"/>
    </f>
    <f name="SUBJ">
     <binary value="true"/>
```

[9] Das Beispiel findet sich auch in unveränderter Form in ISO/DIS 24610-2, S. 28f.

[10] In P5 1.9.1, Kapitel 18, S. 580 wird das am natürlichsprachlichen Beispiel „It is necessary to go“ demonstriert. Um die Person „John“ zum Subjekt dieses Konstrukts zu machen, muss die Konjunktion „for“ eingesetzt werden: „It is necessary *for* John to go“.

```
20     </f>
21    </fs>
22    <then/>
23    <symbol value="for"/>
24   </if>
25  </vDefault>
26 </fDecl>
```

Wie im Listing ersichtlich trennt das leere Element `then` die erfüllte Bedingung und den Wert des Merkmals.

Optionale Merkmalsstruktureinschränkungen können für die Überprüfung auf Validität (vgl. Abschnitt 6.2) relevant sein, denkbar sind hier die bereits erwähnten Co-Constraints. Solche Einschränkungen werden in der Deklaration als Abfolge von `cond`- bzw. `bicond`-Elementen unterhalb des Elements `fsConstraints` realisiert. Die instanziierte Merkmalsstruktur ist nur dann gültig, wenn alle hier aufgeführten Constraints erfüllt sind (P5 1.9.1, Kapitel 18, S. 581). Durch das Element `cond` werden klassische Wenn-Dann-Bedingungen (unidirektionale wechselseitige Einschränkungen) notiert. Listing 6.10 aus P5 1.9.1, Kapitel 18, S. 583 (respektive ISO/DIS 24610-2, S. 32f.) zeigt eine der drei in Gazdar *et al.* (1985, S. 246) aufgeführten Co-Occurrence Constraints, in diesem Fall die Einschränkung, dass eine Phrase (ein Konstrukt mit dem BAR-Wert *1*) kein SUBCAT-Merkmal aufweisen darf.

Listing 6.10: Unidirektionale wechselseitige Einschränkung

```
1  <cond>
2   <fs>
3    <f name="BAR">
4     <symbol value="1"/>
5    </f>
6   </fs>
7   <then/>
8   <fs>
9    <f name="SUBCAT">
10    <binary value="false"/>
11   </f>
12  </fs>
13 </cond>
```

Auch hier wird wieder das leere Element `then` genutzt. Zu beachten ist, dass diese Einschränkung nicht bi-direktional ist (d. h., nicht immer, wenn kein SUBCAT-Merkmal vorliegt, muss es sich um eine Phrase handeln). Für solche wechselseitigen Beziehungen ist das `bicond`-Element zu verwenden. Auch hier dient ein Beispiel aus den TEI GUIDELINES (P5 1.9.1, Kapitel 18, S. 582) zur Erläuterung (Listing 6.11, das Beispiel ist auch zu finden in ISO/DIS 24610-2, S. 32f.).

Listing 6.11: Bidirektionale wechselseitige Einschränkung

```
1  <bicond>
2   <fs>
3    <f name="BAR">
4     <symbol value="0"/>
5    </f>
6   </fs>
7   <iff/>
8   <fs>
9    <f name="N">
10    <binary value="true"/>
```

```
  </f>
  <f name="V">
   <binary value="true"/>
  </f>
  <f name="SUBCAT">
   <binary value="true"/>
  </f>
 </fs>
</bicond>
```

Hiermit wird ausgedrückt, dass ein Konstrukt mit dem BAR-Wert von *0* für die Merkmale N, V und SUBCAT jeweils Werte haben muss – und umgekehrt, d. h., sind für diese Merkmale Werte vorhanden, muss der BAR-Wert *0* sein. Das leere Element `iff` trennt in einer solchen bidirektionalen Einschränkung die Bedingung von der Konsequenz.

6.2.2 Vererbung

Sowohl Merkmalsdeklarationen (durch das Element `fDecl` notiert) als auch Merkmalsstruktureinschränkungen (`fsConstraints`) können mittels Vererbungshierarchien einfacher aufgebaut werden. Ein Beispiel dafür (adaptiert nach ISO/DIS 24610-2, S. 19) zeigt Listing 6.12.

Listing 6.12: Vererbung von Merkmalsdeklarationen

```
<?xml version="1.0" encoding="UTF-8"?>
<fsd>
 <teiHeader>
  <!-- Metadaten -->
 </teiHeader>
 <fsDecl type="sign">
  <fsDescr>The fundamental type for linguistic signs</fsDescr>
  <fDecl name="head">
   <fDescr>Indicates the syntactic head of the sign</fDescr>
   <vRange>
    <fs type="pos"/>
   </vRange>
  </fDecl>
  <fDecl name="spr">
   <fDescr>Indicates the specifiers of the sign</fDescr>
   <vRange>
    <vColl org="list"/>
   </vRange>
  </fDecl>
  <fDecl name="comps">
   <fDescr>Indicates the complements of the sign</fDescr>
   <vRange>
    <vColl org="list"/>
   </vRange>
  </fDecl>
 </fsDecl>
 <fsDecl type="word" baseTypes="sign">
  <fsDescr>The fundamental type for individual words</fsDescr>
  <fDecl name="orth">
   <fDescr>The orthographic representation for this word</fDescr>
   <vRange>
    <string/>
   </vRange>
  </fDecl>
```

```
35  </fsDecl>
36 </fsd>
```

Zunächst wird in Zeile 6 die Merkmalsstruktur SIGN deklariert, die wiederum aus den Merkmalen HEAD, SPR und COMPS besteht. Davon abgeleitet (erkennbar durch das `baseTypes`-Attribut) wird die Merkmalsstruktur WORD deklariert (Zeile 27).[11] Der Effekt dieser Vererbung ist, dass jede Instanziierung der Merkmalsstruktur WORD die Merkmale HEAD, SPR, COMPS sowie zusätzlich ORTH beinhalten muss. Interessant ist die Möglichkeit der Unifikation:

> The attribute baseTypes gives the name of one or more types from which this type inherits feature specifications and constraints; if this type includes a feature specification with the same name as one inherited from any of the types specified by this attribute, or if more than one specification of the same name is inherited, then the possible values of that feature is determined by unification. (P5 1.9.1, Kapitel 18, S. 577).

Ebenso werden Einschränkungen (notiert durch `fsConstraints`) vererbt. Sollte es dabei zu widersprüchlichen Constraints kommen, z. B. durch gegensätzliche Wertebereiche, gibt es keine gültige Merkmalsstruktur dieses Typs (P5 1.9.1, Kapitel 18, S. 577).

6.3 Merkmalsstrukturen zur Darstellung linguistischer Annotationen

Da Merkmalsstrukturen in der Linguistik wie eingangs erwähnt weit verbreitet sind, ist es nicht verwunderlich, dass es eine Reihe linguistischer Anwendungsszenarien gibt. So listet beispielsweise de la Clergerie (2004) eine ganze Reihe an Beispielen auf, die alle aus dem sprachwissenschaftlichen Umfeld stammen und die jeweils als AVM und als TEI-Merkmalsstruktur aufgeführt sind. Ebenfalls erwähnt wurde die Arbeit von Stegmann und Witt (2009), in der die Autoren belegen, dass sich die in diesem Abschnitt angesprochenen XML-Serialisierungen von Merkmalsstrukturen nicht nur zur strukturierten Speicherung von einzelnen Annotationsebenen eignen, sondern auch prinzipiell für multiple Auszeichnungen nutzbar sind. So lassen sich die beiden Annotationen aus Listing 6.13 und 6.14 in die TEI-Merkmalsstruktur aus Listing 6.15 überführen (Beispiele entnommen aus Stegmann und Witt 2009).

Listing 6.13: Morphologische Annotation des Wortes „geben“

```
1 <w>
2  <m type="lexical">geb</m>
3  <m type="flexive">en</m>
4 </w>
```

Dieser Ansatz sieht vor, dass auch die Primärdaten in Form von Merkmalsstrukturen gespeichert werden.[12]

[11] Vererbung auf Basis mehrerer Merkmalsstrukturen ist ebenfalls möglich, in diesem Fall wird die Liste der Basistypen durch Leerzeichen getrennt.

[12] Zu bemängeln ist hierbei, dass die Werte *1*, *2*, ... für die `name`-Attribute der `vLabel`-Elemente nicht

Listing 6.14: Silben-Annotation des Wortes „geben“

```
<w>
 <syll n="s1">ge</syll>
 <syll n="s2">ben</syll>
</w>
```

Listing 6.15: Merkmalsstruktur des Wortes „geben“

```
<?xml version="1.0" encoding="UTF-8"?>
<fs>
 <f name="DATA">
  <vColl org="list">
   <vLabel name="1">
    <string>g</string>
   </vLabel>
   <vLabel name="2">
    <string>e</string>
   </vLabel>
   <vLabel name="3">
    <string>b</string>
   </vLabel>
   <vLabel name="4">
    <string>e</string>
   </vLabel>
   <vLabel name="5">
    <string>n</string>
   </vLabel>
  </vColl>
 </f>
 <f name="DOCUMENTS">
  <vColl org="list">
   <fs>
    <!-- [...] -->
   </fs>
   <fs>
    <!-- [...] -->
   </fs>
  </vColl>
 </f>
</fs>
```

6.4 Beurteilung

Sowohl die in P5 1.9.1, Kapitel 18 als auch in ISO 24610-1:2006 definierten Merkmalsstrukturen eignen sich, nicht nur aufgrund ihres in der Linguistik stark verankerten theoretischen Hintergrunds, durchaus für sprachbezogene Annotationen – allerdings je nach Anwendungszweck mehr oder weniger überzeugend. So lassen sich vorrangig datenzentrierte Informationen wie beispielsweise Lexika recht effizient mittels standardisierter Merkmalsstrukturbeschreibungen kodieren. Die Stärke des Ansatzes liegt sowohl im genutzten formalen Modell des gerichteten, einwurzeligen azyklischen Graphen (vgl. Abschnitt 3.3.3) als auch in der generischen Annotationsstruktur, die

valide sind, da der zugrunde liegende Basisdatentyp `xs:NAME` bzw. `NCName` keine Ziffern als ersten Zeichen erlaubt.

prinzipiell nahezu unbegrenzte Einsatzmöglichkeiten verspricht – auch Dank der Fähigkeit, komplexe Werte in Form von Merkmalsstrukturen rekursiv zu modellieren. Dass diese Kombination realistisch einsetzbar ist, zeigt das GRAPH-BASED ANNOTATION FORMAT (GRAF), dass auf das gleiche formale Modell und Annotationsverfahren setzt (vgl. Abschnitt 9.2). Dem steht entgegen, dass die Notation so allgemein gehalten ist, dass es teilweise nicht zielführend für linguistische Korpora eingesetzt werden kann (vgl. auch die Aussagen von Bański und Przepiórkowski 2010a, in Bezug auf den NCP).

In einer „klassischen" XML-Annotation können Informationen über die Namen der Elemente und Attribute, deren Textinhalte bzw. Werte und die hierarchische Struktur vermittelt werden (vgl. die Ausführungen in Abschnitt 3.1). Bei Verwendung der standardisierten Beschreibung von Merkmalsstrukturen entfällt die Informationsübermittlung über die Namen der Elemente und Attribute, die relevanten Angaben sind in den Werten der Attribute `name` und `value` der jeweiligen Elemente kodiert.[13] Ein positiver Effekt dieser Vorgehensweise ist, dass für die Verarbeitung (Query, Transformation) auch nicht-validierende Prozessoren zum Einsatz kommen können (sofern nicht Integritätsmerkmale oder andere auf eine Dokumentgrammatik angewiesene Eigenschaften genutzt werden), was die Verarbeitung erleichtern kann. Dieser Umstand resultiert daher, dass die Dokumentgrammatik auch Aussagen über die Hierarchie bzw. Verschachtelung der Elemente macht. Diese können für eine optimale Verarbeitung von Instanzen dieser Auszeichnungssprache essentiell sein. Im Falle der standardisierten Beschreibung von Merkmalsstrukturen fehlen diese Informationen, da hier eine geringe Anzahl von Elementen (vorrangig `fs` und `f`) rekursiv verschachtelt werden kann. Natürlich kann mittels rekursiver Verarbeitung dieses berücksichtigt werden, aber prinzipiell muss in Abhängigkeit der Werte der Instanz vorgegangen werden, was die Möglichkeiten einschränkt. Insofern kann zumindest angezweifelt werden, inwieweit die Restriktion der Generic Identifier den Austausch und die Verarbeitung linguistischer Daten erheblich vereinfacht – Schlüsselelement ist hier sicherlich die einheitliche Verwendung von Datenkategorien (vgl. Kapitel 8).

Die Nutzung von RELAX NG als Constraint Language zur Definition der Serialisierung standardisierter Merkmalsstrukturbeschreibungen ist mit keinen besonderen Vorteilen verbunden, da die formale Ausdrucksstärke des Grammatikformalismus' nicht ausgereizt wird. Die Dokumentgrammatik entspricht in dieser Hinsicht einer LTG, so dass zumindest aus formalen Gesichtspunkten auch DTD oder XSD als Alternative zur Verfügung stünde.[14] Allerdings sind internationale Normen dazu angehalten, sich auf bestehende Standards zu beziehen. Da RELAX NG ebenfalls als Internationaler Standard veröffentlicht ist, ist die Nutzung nahe liegend.[15]

Da die Spezifikation keine Vorgaben über konkrete Ausprägungen der Merkmale und

[13] Anzumerken ist an dieser Stelle natürlich, dass bei atomaren Merkmalen durch Kindelemente wie `binary` oder `symbol` durchaus Aussagen über den Elementnamen getroffen werden, deren Informationsgehalt ist aber sehr begrenzt.

[14] Die TEI P5, die ebenfalls die Merkmalsstrukturbeschreibungen enthält, ist auch als mittels dieser beiden Constraint Languages formalisierten Dokumentgrammatik verfügbar (vgl. Abschnitt 5.9).

[15] Zu beachten ist, dass sowohl die XML-Syntax als auch die Compact Syntax von RELAX NG im normativen Teil von ISO/IEC 19757-2:2003 enthalten sind, während RELAX NG-Prozessoren nur die XML-Syntax unterstützen müssen.

deren Werte macht, erlauben die hier diskutierten standardisierten Merkmalsstrukturbeschreibungen die Trennung von Level und Layer (vgl. Abschnitt 3.5), da die konzeptuelle Ebene (der Level) durch die Bezeichner der Merkmale und Werte repräsentiert wird. Auch sind problemlos komplexe Merkmalsstrukturen möglich, die alle dasselbe konzeptuelle Merkmal auf unterschiedliche Art (also mit verschiedenen Merkmalsnamen und Werten) beschreiben.

Die Einschätzung einer nachhaltigen Verwendbarkeit der Normen ISO 24610-1:2006 und ISO 24610-2:2011 ist abschließend nicht ganz einfach, da die Unterschiede zur TEI-Version gering sind. Für diese spricht allerdings die Tatsache, dass die TEI GUIDELINES über Jahrzehnte hinweg aktiv von einem internationalen Konsortium gepflegt und weiterentwickelt wurden und werden. Zwar können auch internationale Normen überarbeitet werden, die Erfahrung zeigt allerdings, dass diese Möglichkeit eher selten genutzt wird. Im Gegensatz zu ISO 24610-1:2006 sind die TEI GUIDELINES darüber hinaus unter einer offenen Lizenz frei verfügbar, was sowohl den Aspekt der Kosten als auch den der Adaptierbarkeit einschließt. Mit ROMA (vgl. Abschnitt 5.2) steht ein mächtiges und einfach zu nutzendes Werkzeug zur Verfügung, um Anpassungen an die TEI (und damit auch TEI-Merkmalsstrukturen) vorzunehmen, was ebenfalls einen Vorteil der TEI-Variante darstellt.

7

CES und XCES – der Corpus Encoding Standard?

Im Februar 1993 wurde unter Federführung des *Linguistic Research and Engineering (LRE)*-Programms das Projekt LRE-61-100, „Expert Advisory Group on Language Engineering Standards“ (EAGLES) als Initiative der Europäischen Kommission aus der Taufe gehoben.[1] Das Projekt sollte die Bereitstellung von standardisierten Lösungen für die Speicherung von großen Sprachressourcen (darunter Sprach- und Textkorpora) und deren Verarbeitung durch computerlinguistische Verfahren beschleunigen. Ein weiterer Aspekt war die Evaluation von Ressourcen und Werkzeugen. Als Ergebnis wurden die EAGLES GUIDELINES erarbeitet, eine Ansammlung an Empfehlungen für De-facto-Standards und *Good Practice* im Bereich der Sprachverarbeitung, angelehnt an die TEI GUIDELINES. Das EAGLES-Projekt war unterteilt in fünf Arbeitsgruppen:

- WG 1 „Text Corpora“
- WG 2 „Computational Lexicons“
- WG 3 „Grammar Formalism“
- WG 4 „Evaluation“
- WG 5 „Spoken Language“

An dieser Stelle soll es vorrangig um die Arbeiten der Arbeitsgruppe 1 (WG 1) gehen, da die hier erarbeitete Spezifikation primär zur Speicherung und Strukturierung von Textkorpusdaten eingesetzt wird. Teil der EAGLES GUIDELINES ist der CORPUS ENCODING STANDARD (CES). Wie der Name bereits suggeriert, wurde CES zur Annotation und Speicherung von Korpora in NLP-Anwendungen konzipiert und stützte sich dabei

[1] Weitere Informationen unter `http://www.ilc.cnr.it/EAGLES96/home.html`, zuletzt abgerufen am 19.04.2012.

zunächst auf SGML und die TEI GUIDELINES.[2] Begründet wurde die Erarbeitung der Spezifikation mit der gesteigerten Nachfrage nach großen Korpora, die in den 90er Jahren des letzten Jahrhunderts durch das Wiedererstarken empirischer Methoden in der Sprachverarbeitung hervorgerufen wurde. Auf der ganzen Welt wurden (und werden) Anstrengungen unternommen, große Sprachkorpora zu erfassen, darunter solch ambitionierte Projekte wie der „British National Corpus" (BNC) oder die *European Language Resources Association* (ELRA, vgl. Ide, Priest-Dorman *et al.* 1996).[3]

Der CORPUS ENCODING STANDARD wurde in Kooperation mit dem *Vassar College* (New York) und dem *Centre National de la Recherche Scientifique* (*CNRS*) in Aix-en-Provence (im Folgenden als Vassar/CNRS zusammengefasst) maßgeblich von den europäischen Projekten „MULTEXT"[4] und „EAGLES" entwickelt. Die Spezifikation ist eine Anwendung der TEI GUIDELINES (vgl. Kapitel 5) in der zu diesem Zeitpunkt aktuellen Version P3 (Anhang 7: „How to use the CES" in Ide, Priest-Dorman *et al.* 1996, beschreibt die Verwendung als TEI-Erweiterung sowie als eigenständige Dokumentgrammatik). Ziel ist die Realisierung eines minimalen TEI-konformen Annotationsstandards für strukturelle und linguistische Informationen im Rahmen von Korpusstudien. Als Korpus im Sinne des CES wird eine Sammlung von linguistischen Daten (sowohl gesprochene als auch geschriebene Sprache) in einer oder mehr Sprachen gesehen, die aufgrund linguistischer Kriterien gesammelt wurden. Diese müssen allerdings nicht bestimmten Design-Kriterien unterliegen und können somit alle Arten von Texten (Texttypen) enthalten (Ide, Priest-Dorman *et al.* 1996, Abschnitt 0.2.1, „Text Types").

Die Spezifikation besteht aus acht Teilen und zehn Anhängen, sowie den entsprechenden Dokumentgrammatiken und ist online verfügbar.[5] Teil 0, „Introduction", führt in die Hintergründe ein und definiert den Skopus der Spezifikation und die oben genannte Korpusdefinition. Teil 1, „General Principles", listet einige weitere Definitionen und die durch den CES angestrebten Ebenen der Standardisierung auf. Ebenfalls hier enthalten ist die Abgrenzung zu den TEI GUIDELINES (Abschnitt 1.6, „Customization of the TEI"), die in Form von drei Anpassungsdateien (*Customization Files*) bzw. durch die allein lauffähigen Dokumentgrammatiken (in Form von SGML-DTDs) verfügbar sind.[6] Hier wird auch die Umbenennung der drei P3-Elemente `TEI.2`, `teiCorpus.2` und `teiHeader` in `cesDoc`, `cesCorpus` und `cesHeader` kurz thematisiert. Teil 2, „Recommendations common to all documents", enthält allgemeine Richtlinien bzgl. der verwendeten Metasprache SGML und des Zeichensatzes. Einzelne Abschnitte dieses Teils sind nicht fertiggestellt und mit dem Hinweis „[THIS SECTION IS UNDER DEVELOPMENT]" versehen. Teil 3, „The Header", stellt die CES-Metadaten vor (vgl. Abschnitt 7.3 in diesem Kapitel), die sich am TEI-Header orientieren. Teil 4, „Encoding Primary Data", beinhaltet

[2] Der Zusammenhang zwischen dem SGML-basierten CES und dem mittels XML definierten XCES wird in Abschnitt 7.1 thematisiert.

[3] Die ELRA ist auch bekannt als Ausrichter der Konferenz „International Language Resources and Evaluation" (LREC). Weitere Informationen zu den Zielen der Organisation, zu denen neben der Standardisierung auch die Verbreitung linguistischer Korpora gehört, finden sich unter der Adresse `http://www.elra.info/`, zuletzt abgerufen am 19.04.2012.

[4] Projekt-Homepage unter `http://www.lpl.univ-aix.fr/projects/multext/`, zuletzt abgerufen am 19.04.2012.

[5] Einsehbar unter `http://www.cs.vassar.edu/CES/`, zuletzt abgerufen am 19.04.2012.

[6] Hierzu sei ergänzend auf Ide, Priest-Dorman *et al.* (1996, Anhang 7, „How to use the CES") verwiesen.

die grundlegende Annotation von Primärdaten, die in Abschnitt 7.2 näher erläutert wird. Teil 5, „Encoding linguistic annotation“, stellt die grundlegenden CES-Prinzipien zur Annotation linguistischer Korpora, vorrangig die Standoff-Annotation (vgl. Abschnitt 3.3.2.3) und die dazu notwendigen Zeiger, sowie das Annotationsinventar der `cesAna`-Dokumentgrammatik vor (vgl. Abschnitte 7.4 und 7.5 in diesem Kapitel). Die Teile 6, „Encoding speech“, und 7, „Encoding linguistic annotation for speech“, sind nur als Platzhalter mit dem Vermerk „[under construction]“ vorhanden. Da die letzten Änderungen am Text 1996 erfolgten, ist mit keiner Überarbeitung bzw. Erweiterung zu rechnen. Die zehn Anhänge beinhalten sowohl Verweise auf andere Spezifikationen und Standards bzw. Literatur und URLs als auch einen Index der Elemente, die Dokumentgrammatiken und Hinweise zur Anwendung.

Im Gegensatz zu den TEI GUIDELINES, die eine Klassifikation von Texten primär anhand ihrer externen (und damit nicht linguistisch motivierten) Merkmale festmachen, berücksichtigt CES zusätzliche linguistische Merkmale, um Texte zu klassifizieren und zu strukturieren und nutzt zu diesem Zweck den modularen Aufbau der TEI, der es ermöglicht, sich aus einem Inventar aus den zur Verfügung stehenden Teil-Dokumentgrammatiken zu bedienen. Damit schränkt die Spezifikation die Zielmenge der damit zu beschreibenden Dokumente gegenüber den TEI GUIDELINES ein, erlaubt im Gegenzug aber die genauere Dokumentation linguistischer Phänomene (Sperberg-McQueen 2011). Darüber hinaus erweitert CES die P3 um linguistisch motivierte Auszeichnungen (z. B. um Vorgaben zur morphosyntaktischen Annotation in der `cesAna`-Dokumentgrammatik, Ide 1998). Bański (2001, S. 4) fasst das Verhältnis zwischen CES und TEI wie folgt zusammen:

> The original, SGML-based CES can be thought of as an application of the TEI in that it was created by adding the appropriate set of DTD fragments as so-called extension files to the main TEI DTD. Because the CES targets language corpora and their applications in language engineering, on the one hand, it limits the power of the original TEI scheme and on the other, it adds extra features necessary for the proper handling of corpora.

Ein Beispiel für die Einschränkung der Annotationsmöglichkeiten ist, dass der zu annotierende Text als Inhalt von Elementen (also nicht in Form von Attributwerten) adressierbar sein soll. Weiterhin sollen Elemente nicht verschachtelt und Zusatzinformationen (wie Hervorhebungen) über Attribute gespeichert werden (Bański 2001, S. 6). Damit soll generell die Verarbeitung vereinfacht und beschleunigt werden. Weitere Ziele sind die maximale Nutzbarkeit und Nachhaltigkeit der annotierten Daten, sowie Kosteneffizienz. Beide Punkte resultieren aus der Tatsache, dass Primärdaten bereits Annotationen enthalten können, deren Umwandlung und Entfernung aufwendig und damit kostenintensiv sein kann (Ide 1998).

7.1 CES und XCES

Bereits vor der Verabschiedung der TEI P4 und der damit verbundenen Umstellung auf die Metasprache XML, wurde mit XCES eine ebenfalls auf XML basierende Version des

Corpus Encoding Standards vorgestellt (vgl. Ide, Bonhomme *et al.* 2000). Die Frage, inwieweit XCES eine kompatible Modifikation der P4 (im Sinne von *TEI-compliant*) ist, ist tendenziell zu verneinen. Dagegen spricht, dass die vorgenommenen Änderungen nicht dokumentiert und somit nicht vollständig nachvollziehbar durchgeführt wurden. Diese Analyse wird noch dadurch erschwert, dass es zwei verschiedenen XCES-Fassungen gibt: Eine als DTD formalisierte Reformulierung von CES und eine aktuellere mittels XML Schema definierte Version, ein Umstand, der auch in Bański und Przepiórkowski (2010a, S. 100, Fußnote 2) bemängelt wird: „Two different sets of schemata have co-existed on XCES WWW pages since 2003, one given as DTD, another as XML Schema, without any clear indication that they specify different structures." Die erste Fassung ist eine durch Entfernen von vorhandenen Inklusionen bzw. Exklusionen (vgl. Abschnitt 3.2) und Anpassungen in Bezug auf gemischte Inhaltsmodelle XML-konform modifizierte CES-DTD. Die spätere XCES-Fassung führt XML-Namensräume ein und verwendet XML Schema als normative Constraint Language (Ide, Bonhomme *et al.* 2000, S. 825f.). XPath, XLink und die XML Pointer Language (XPointer, DeRose, Daniel, Grosso *et al.* 2002) werden zur Verknüpfung von Annotation und Primärdatum eingesetzt, da der XML-inhärente Integritätsmechanismus per `ID/IDREF/IDREFS` nur innerhalb einer Instanz anwendbar ist und der in CES vorhandene, auf HyTime (ISO/IEC 10744:1997; Goldfarb, Newcomb *et al.* 1997) und den Extended Pointer der TEI Guidelines basierende Ansatz in XML nicht zur Verfügung steht (vgl. auch die Ausführungen in Abschnitt 7.4).

Neben der bereits erwähnten unterschiedlich aufgebauten `cesAna`-Dokumentgrammatik besteht ein weiterer Unterschied zwischen CES und XCES darin, dass in Ide, Priest-Dorman *et al.* (1996) für die Annotation gesprochener Sprache weder Auszeichnungsinventar noch Dokumentation vorhanden ist (die entsprechenden Teile 6 und 7, „Encoding speech" bzw. „Encoding linguistic annotation for speech", sind mit dem Vermerk „nder construction" versehen). Dagegen enthält die aktuelle Fassung von XCES mit `xcesSpoken.xsd` ein XML-Schema, das als Modifikation von `xces.Doc.xsd` die grundlegende Strukturierung gesprochener Sprache erlaubt.

Im weiteren Verlauf dieses Kapitels werden bei der Diskussion der Spezifikation Unterschiede zwischen den beiden Fassungen explizit angesprochen, ansonsten ist der Term „CES" als Sammelbegriff zu verstehen. Auf Ebene der Dokumentgrammatik setzt sich der Corpus Encoding Standard aus drei Dokumentgrammatiken zusammen: `cesDoc`, `cesAna` und `cesAlign`. Für CES sind diese in Form von SGML-DTDs formalisiert, für XCES sind sowohl XML-Schemata als auch XML-DTDs vorhanden – wobei auf Abweichungen zwischen beiden letzteren Formalisierungen im Folgenden gesondert hingewiesen wird.

7.2 cesDoc: Strukturierung der Primärdaten

Die Spezifikation sieht drei mögliche Ebenen einer CES-konformen Kodierung der Primärdaten vor (Ide, Priest-Dorman *et al.* 1996, Teil 4, „Encoding Primary Data"; Ide 1998). Eine Datei, die gegen `cesDoc` validiert, enthält die Primärdaten sowie eine minimale Segmentierung in Form der logischen Dokumentstruktur (bis auf Absatzebene) – die

Instanz ist damit CES-Level 1-konform.[7] Zusätzlich sind Metadaten in Form des `cesHeader` (vgl. Abschnitt 7.3) vorgesehen, die Auskunft über alle zur Annotation verwendeten Elemente und Attribute geben. Zur Vereinfachung reicht es, wenn Absatzgrenzen durch das CES-eigene `p` ausgezeichnet werden, darüber hinaus müssen keine Angaben (z. B. bzgl. Listenpunkte o. ä. gemacht werden). Unterhalb der Absatzebene sollten Hervorhebungen durch das generische `hi` (*highlighted*, hervorgehoben) ersetzt werden, so dass so wenig Information wie möglich durch die Umwandlung verloren wird.

Level 2-konforme CES-Instanzen enthalten zusätzlich spezifische Angaben über den Grund der Hervorhebungen innerhalb von Absätzen (d. h., `hi`-Elemente sollten durch präzisere Auszeichnungen ersetzt werden). Darüber hinaus muss die Annotation eines Phänomens einheitlich sein, d. h., alle Vorkommen müssen durch die gleiche Auszeichnung im Text markiert sein. Diesem Punkt, der in Ide (1998, S. 464) als Prozess der *Up-translation* genannt wird, und die Ersetzung impliziter Auszeichnungen (z. B. durch Formatierungen im Text) durch deskriptive Annotationen bezeichnet, wird in CES erstmalig erhöhte Aufmerksamkeit gewidmet.[8] Entitäten müssen in CES SGML- bzw. XML-konform sein (inkl. Anführungszeichen, die entweder durch entsprechende Entitäten oder `q`- bzw. `quote`-Elemente ersetzt werden müssen), Listen werden als solche ausgezeichnet.

Instanzen, die Level 3 genügen, unterliegen folgenden zusätzlichen Constraints:

- Abkürzungen, Nummern, Fremdwörter und -Phrasen sind entsprechend ausgezeichnet;
- Morphosyntaktische Annotation durch ein vom Benutzer vorgegebenes Tagset;
- Sätze und wörtliche Rede sind durch die entsprechenden Elemente `s` und `q` ausgezeichnet.

Das `text`-Element, das als Geschwisterelement des Headers angelegt ist, kann unterhalb von `body` allgemeine Unterteilungen (`div`, auch rekursiv verschachtelbar) oder absatzbildende Elemente enthalten. Alternativ kann wie in den TEI GUIDELINES auch das `group`-Element als Wrapper für mehrere `body`-Vorkommen genutzt werden. `div`-Elemente können durch die Elemente `opener`, `byline`, `closener` und `endByline` mit weiteren Informationen (wie z. B. einer Datumsangabe oder Schlüsselwörtern in den Elementen `opener` und `closer` oder einer Autorenkennung in der `byline` bzw. der `endByline`) versehen werden. In CES ist zusätzlich das `head`-Element an dieser Stelle enthalten, das in XCES zu der Gruppe der absatzbildenden Elemente gezählt wird. Weitere Mitglieder dieser Gruppe sind allgemeine Absätze (`p`), Segmente gesprochener Sprache (`sp`), Über- bzw. Unterschriften (`caption`), Zitate (`quote`), Gedichte (`poem`), Listen (`list`), Abbildungen (`figure`), bibliographische Referenzen (`bibl`), Hinweise (`note`) oder Tabellen (`table`). Für den Aufbau der jeweiligen Elemente bzgl. Kindelemente und Attribute sei auf Ide, Priest-Dorman *et al.* (1996) verwiesen.

[7] Folgerichtig wird auf `http://www.xces.org/dtds.html` die Dokumentgrammatik `xcesDoc.dtd` als „Encoding conventions for level 1“ bezeichnet, zuletzt abgerufen am 19.04.2012.

[8] Eine aktuell gebräuchlichere Bezeichnung wäre *Up-Conversion*, so z. B. in Kay (2004); Haupt und Stührenberg (2010).

Unterhalb der Absatzgrenze (oder alternativ direkt unterhalb des body-Elements) ist eine Anzahl von aus den TEI GUIDELINES stammenden Elementen, darunter Abkürzungen (abbr), Datums-, Zeit- und Maßangaben (date, time, measure), Nummern (num), Eigennamen (name), technische Begriffe (term) oder Token (token), aber auch allgemeinere Hervorhebungen (das bereits erwähnte hi) oder Betonungen (emph) erlaubt. Diese werden in Ide, Priest-Dorman *et al.* (1996, Abschnitt 4.5.9, „Sub-paragraph (phrase-level) elements“) in linguistisch motivierte Auszeichnungen, redaktionelle Überarbeitungen gegenüber dem Originaltext, die bereits genannten Hervorhebungen, orthographische Sätze (oder sonstige *S-Units*) und wörtliche Rede, sowie Referenzen und Zeiger unterteilt. Auch hier sei auf den entsprechenden Abschnitt der Spezifikation verwiesen (die technisch motivierten Änderungen in Bezug auf Zeiger werden in Abschnitt 7.4 auf Seite 170 diskutiert).

Auch wenn Standoff-Annotation das bevorzugte Mittel der Wahl ist, enthält der CORPUS ENCODING STANDARD ein Tagset zur Inline-Annotation morphosyntaktischer Phänomene (Ide, Priest-Dorman *et al.* 1996, Abschnitt 4.5.13, „Encoding morpho-syntactic annotation in the primary data“; Ide 1998, S. 467). Zu diesem Zweck kommt ein Modul mit den entsprechenden Elementen und Attributen der cesAna-Dokumentgrammatik zum Einsatz. Alternativ dazu kann das Annotationsinventar einer externen Dokumentgrammatik herangezogen werden.

Listing 7.1 zeigt eine manuell erstellte Primärdatendatei, konform nach XCES-Level 1. Grundlage ist die aus dem „Open American National Corpus“ (OANC, vgl. Abschnitt 7.8) entnommene Datei TheStory.txt.[9]

Listing 7.1: XCES-Primärdaten in Level 1-Notation

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cesDoc xmlns="http://www.xces.org/schema/2003"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.xces.org/schema/2003 xcesDoc.xsd" version="0.4">
5   <text>
6    <body>
7     <p>The Story Continues . . . a serial enovel by Ferd Eggan</p>
8     <p>1 Welcome to Hotel Real Desert</p>
9     <p>But he never fell into the error of arresting his intellectual development by any formal
10     acceptance of creed or system, or of mistaking, for a house in which to live, an inn that is but
11     suitable for the sojourn of a night in which there are no stars and the moon is in travail</p>
12    <p>The Hotel</p>
13    <p>Hotel is next door to a perfect metaphor for the mind, and thus for psychoanalysis. In my
14     father's house are many mansions?To get there you have to leave somewhere else...</p>
15    <!-- [...] -->
16   </body>
17  </text>
18 </cesDoc>
```

Wie auch Bański (2001, S. 4) konstatiert, spiegeln die drei Konformitäts-Ebenen die Akquise von Primärdaten wider, deren Annotation nach und nach ergänzt wird. So lässt

[9] Die Datei ist nach Download des OANC unter der URL http://anc.org/OANC/OANC-1.0.1-UTF8.zip und Entpacken des Zip-Archivs in der Verzeichnisstruktur zu finden unter data/written_1/fiction/eggan als TheStory.txt. Zu beachten ist, dass hierzu der OANC in der vorherigen Version („Open ANC in the original XML format“) herunterzuladen ist – weitere Ausführungen dazu im Abschnitt 7.8. Zuletzt abgerufen am 19.04.2012.

sich Level 1-Konformität oftmals durch geringen Aufwand und eine semi-automatische Verarbeitung der Primärdaten erreichen, während höhere Ebenen zusätzlichen Einsatz erfordern. Dementsprechend konzentriert sich das Annotationsinventar auf drei Bereiche:

1. Markup für das Gesamtdokument
   Bibliographische Informationen, Kodierungen, etc. – vorrangig Metadaten.

2. Größere strukturelle Einheiten
   Bände, Kapitel, Abschnitte, etc. bis auf Absatzgrenzen. Zusätzlich Abbildungen, Titel, Tabellen u. a..

3. Auszeichnungen unterhalb der Absatzebene
   Sätze, wörtliche Rede, Wörter, Abkürzungen, etc.

7.3 cesHeader: Metadaten

Metadaten werden im `cesHeader` strukturiert gespeichert, der analog zum TEI-Header Informationen über das Dokument, die Enkodierung und die bisherige Bearbeitung beinhalten kann. Hier kann auch der entsprechende Level in Form des gleichnamigen Attributs des `conformance`-Elements kodiert werden (Vorgabewert ist hier *1*). Während in Ide, Priest-Dorman *et al.* 1996 Metadaten in allen Dateien vorgesehen waren, plädiert XCES dafür, diese in eine externe Instanz auszulagern – als Folge ist das Vorkommen des `cesHeader`-Elements in den XML-Schemata als optional (`minOccurs="0"`) deklariert. Sinnvollerweise sind die Elemente `header` und `cesHeader` aus dem XCES-Namensraum, die beide alternativ genutzt werden können, in einem eigenen XSD (`xcesHeader.xsd`) deklariert. Listing 7.2 zeigt die Metadaten für eine XCES-Instanz aus dem OANC.[10]

[10] Die Datei ist nach Download des OANC unter der URL `http://anc.org/OANC/OANC-1.0.1-UTF8.zip` und Entpacken des Zip-Archivs in der Verzeichnisstruktur zu finden unter `data/written_1/fiction/eggan` als `TheStory.anc`, zuletzt abgerufen am 19.04.2012.

Listing 7.2: XCES-Header

```
<?xml version="1.0" encoding="UTF-8"?>
<cesHeader creator="KBS" date.created="20050222" version="1.0.4" xmlns="http://www.xces.org/schema/2003">
 <fileDesc>
  <titleStmt>
   <title>The Story Continues</title>
   <author>Ferd Eggan</author>
  </titleStmt>
  <sourceDesc>
   <biblStruct>
    <monogr>
     <imprint>
      <publisher>Ferd Eggan</publisher>
      <pubDate>2005</pubDate>
      <eAddress type="url">http://www.ferdeggan.com</eAddress>
     </imprint>
    </monogr>
   </biblStruct>
  </sourceDesc>
 </fileDesc>
 <profileDesc>
  <textClass>
   <domain>Fiction</domain>
   <subdomain>General fiction</subdomain>
   <subject>Misc.</subject>
   <audience>adult</audience>
   <medium>online novel</medium>
  </textClass>
  <annotations>
   <annotation ann.loc="TheStory.txt" type="content">Text content</annotation>
   <annotation ann.loc="TheStory-logical.xml" type="logical">Logical structure</annotation>
   <annotation ann.loc="TheStory-s.xml" type="s">Sentence boundaries</annotation>
   <annotation ann.loc="TheStory-hepple.xml" type="hepple">Hepple part of speech tags</annotation>
   <annotation ann.loc="TheStory-np.xml" type="np">Noun chunks</annotation>
   <annotation ann.loc="TheStory-vp.xml" type="vp">Verb chunks</annotation>
  </annotations>
 </profileDesc>
</cesHeader>
```

7.4 Zeiger in XCES

Eine Verknüpfung zwischen Standoff-Annotation und annotierten Daten wird in CES über den TEI-eigenen, auf HyTime basierenden Linking-Mechanismus realisiert. Da sich HyTime nicht durchsetzen konnte, lehnt sich das Linking in der aktuellen Version P5 an etabliertere Standards wie XLink und XPointer an. Einen ähnlichen Weg geht XCES: Hier wird ebenfalls XLink und XPointer eingesetzt. Das Listing 7.3 zeigt einen Zeiger, wie er mittels HyTime/TEI Extended Pointer realisiert wird, während Listing 7.4 die davon abgeleitete verkürzte CES-Variante demonstriert. Listing 7.5 schließlich zeigt die XCES-Variante (alle Beispiele aus Ide, Bonhomme *et al.* 2000, S. 827).

Listing 7.3: HyTime/TEI P3-Extended Pointer

```
<tok from="CHILD (1) (2) STRLOC (10)" to="CHILD (1) (2) STRLOC (22)">
```

Listing 7.4: CES-Pointer in der verkürzten Version

```
1 <tok from="1.2\10" to="1.2\22">
```

Listing 7.5: XCES-Pointer

```
1 <tok xlink:href= "http://www.loria.fr/doc.xml#xptr (substring(/p/s[2]/text(), 10, 12))">
```

In allen Fällen wird jeweils auf das zweite Kindelement unterhalb des Wurzelelements zugegriffen und von dessen Textinhalt (dessen Textknoten) eine Teilzeichenkette (beginnend bei Zeichen 10 bis hin zum Zeichen 22) ausgewählt. In XCES wird zusätzlich noch die Instanz, auf deren Inhalt der Zeiger verweist, angegeben. Mittels `xml:base`-Attribut kann diese Angabe verkürzt werden, wie Listing 7.6 zeigt.

Listing 7.6: XCES-Pointer mit `xml:base`-Angabe

```
1 <chunk xml:base="http://www.loria.fr/doc.xml#">
2  <tok xlink:href="xptr(substring (/p/s[2]/text(), 10, 12))"/>
3  <tok xlink:href="xptr(substring (/p/s[2]/text(), 24, 4))"/>
4 </chunk>
```

Problematisch an dieser Stelle ist, dass die von XCES propagierte XML Pointer Language durch das XPointer Framework bzw. das XPointer xpointer() Scheme ersetzt wurde, womit die Schreibweise der in Ide, Bonhomme *et al.* (2000) enthaltenen Beispiele nicht mehr korrekt ist. Ein valides Beispiel ist in Listing 7.8 auf Seite 174 aufgeführt. Allerdings ist anzumerken, dass selbst das als Nachfolger propagierte XPointer xpointer() Scheme weiterhin nur den Status eines Working Draft hat und nur wenige Implementierungen alle vier zu XPointer gehörenden Spezifikationen unterstützen (bestehend aus dem Framework, dem xpointer() Scheme, dem element() Scheme und dem xmlns() Schema).

7.5 cesAna: Annotationsebenen

Die Dokumentgramamtik `cesAna` dient ebenfalls der Segmentierung und zusätzlich der grammatikalischen Annotation, inkl. Satzgrenzen und Token (mit orthographischer Form, Lemma und morphosyntaktischer Spezifikation).

Zu beachten sind hier Unterschiede zwischen CES und XCES: Während die `cesAna.dtd` unterhalb des `cesAna`-Wurzelelements einen optionalen Header und ein obligatorisches `chunkList`-Element vorsieht (wobei letzteres eine Anzahl an `chunk`-Elementen, bestehend aus `tok`, `s` oder `par`, beinhaltet), ist in der XSD-Variante `cesAna.xsd` direkt unterhalb von `cesAna` eine beliebige Anzahl von `struct`-Elementen erlaubt (und nur diese). Unterhalb davon können `feat`-Elemente Merkmalsstrukturen abbilden (vgl. Kapitel 6). Damit folgt die XCES-Variante stringent der Standoff-Systematik, wie auch das Listing 7.7 verdeutlicht, das einen Ausschnitt aus der Instanz `TheStory-logical.xml` aus dem OANC zeigt.[11]

[11] Die Datei ist nach Download des OANC unter der URL `http://anc.org/OANC-1.0.1-UTF8.zip` und Entpacken des Zip-Archivs in der Verzeichnisstruktur zu finden unter `data/written_1/fiction/eggan` als `TheStory-logical.xml`. Zu beachten ist, dass hierzu der OANC in der vorherigen Version („Open ANC in the original XML format“) herunterzuladen ist – weitere Ausführungen dazu im Abschnitt 7.8. Zuletzt abgerufen am 19.04.2012.

Listing 7.7: XCES-Instanz aus dem OANC

```
<?xml version="1.0" encoding="UTF-8"?>
<cesAna xmlns="http://www.xces.org/schema/2003" version="1.0.4">
 <struct type="cesDoc" from="0" to="400307">
  <feat name="xmlns" value="http://www.xces.org/schema/2003"/>
  <feat name="version" value="1.0.4"/>
 </struct>
 <struct type="text" from="2" to="400306"/>
 <struct type="body" from="5" to="400304"/>
 <struct type="div" from="9" to="73"/>
 <struct type="p" from="14" to="69">
  <feat name="id" value="p1"/>
 </struct>
 <struct type="head" from="77" to="108">
  <feat name="type" value="h1"/>
 </struct>
 <struct type="p" from="112" to="414">
  <feat name="id" value="p2"/>
 </struct>
 <struct type="hi" from="409" to="410">
  <feat name="rend" value="sup"/>
 </struct>
 <struct type="head" from="418" to="427">
  <feat name="type" value="h1"/>
 </struct>
 <!-- [...] -->
</cesAna>
```

Diese Form der Annotation geht zurück auf die Überlegungen zum GENERIC MAPPING TOOL (vgl. auch Abschnitt 4.2). Der Aufbau unterscheidet sich dabei von den Merkmalsstrukturbeschreibungen, wie sie in Kapitel 6 vorgestellt werden, derart, dass Baum-Hierarchien und Knoten-Annotationen kombiniert werden. Das Element `struct`[12] repräsentiert den Knoten im Baum und kann rekursiv verschachtelt werden, um die Baumstruktur abzubilden. Zulässige Attribute sind `type`, um die Art der Annotation zu spezifizieren (etwa vergleichbar mit der konzeptuellen Ebene aus Abschnitt 3.5 dieser Arbeit), `ID` als eindeutiger Bezeichner eines Knotens, und `ref`, sofern dieser Knoten einen anderen repräsentiert (als implizite Struktur, vergleichbar mit Strukturteilung in Merkmalsstrukturen). XCES erweitert die Anzahl der Attribute gegenüber GMT, beispielsweise um das generische `xml:lang`, `orig`, `reg` oder `affix`. Ebenfalls in GMT noch nicht vorhanden sind die Attribute `from` und `to`, die mittels Ganzzahlen (Datentyp `xs:int`) Angaben über den Bereich der Primärdaten machen, der annotiert wird.[13] Unterhalb von `struct` können `feat`-Elemente Informationen über den entsprechenden Knoten speichern (in Romary 2001, wird das Element als Träger einer Datenkategorie bezeichnet). In GMT können diese rekursiv verschachtelt sein, um komplexe Merkmalsstrukturen abzubilden, das aktuelle XCES-XML-Schema erlaubt dies nicht mehr. Diese Verlagerung hin von einem verbindlichen Tagset zur Repräsentation morphosyntaktischer Annotationen zu einer allgemeineren Merkmalsstruktur wird von Bański und Przepiórkowski (2010a, S. 98) kritisiert:

> A new – more abstract – version of XCES was introduced around 2003, where

[12] In der ersten Darstellung in Ide, Kilgarriff *et al.* (2000) trägt das Element noch den Namen `struc`.

[13] Eine Aussage die Einheit betreffend (z. B. Zeichen, Byte, etc.) fehlt, da das Schema nicht dokumentiert ist.

> [the] concrete morphosyntactic schema was replaced by a general feature structure mechanism, different from the ISO Feature Structure Representation (FSR) standard (ISO 24610-1). In our view, this is a step back, as adopting a more abstract representation requires more work on the part of corpus developers.

Zusätzlich problematisch ist an dieser Stelle, dass zu XCES keine aktuelle Dokumentation existiert. Die Webseite `http://www.xces.org/` verweist auf Ide, Priest-Dorman *et al.* (1996) – da es aber Abweichungen zwischen beiden Spezifikationen gibt, und auch die XSD-Dateien kaum Kommentare aufweisen, sind definitive Aussagen nicht immer möglich. Auch dieser Aspekt wird in Bański und Przepiórkowski (2010a, S. 98) entsprechend negativ gewürdigt.

7.6 cesAlign: Verknüpfungen

Die Dokumentgrammatik `cesAlign` definiert eine Instanz zur Verknüpfung von (X)CES-Annotationen zueinander bzw. zur Verknüpfung von Primärdaten und Annotationen. Eine nach dieser Dokumentgrammatik valide Instanz besteht neben einem optionalen `cesHeader`-Element aus genau einem `linkList`-Element, welches wiederum Vorkommen von `linkGrp`-Elementen beinhaltet. Das `cesAlign`-Wurzelelement kann über Attribute weiter spezifiziert werden. Das `type`-Attribut kann in der CES-Variante die Werte *PAR*, *SENT* oder *TOK* annehmen, was für Alignierung über Absätze, Sätze respektive Token steht. Dieses Attribut ist in der XCES-Version nicht mehr enthalten. In beiden Fassungen vorhanden sind aber die Attribute `fromDoc`, `toDoc` und `version`. Während letzteres dem Abgleich der genutzten Version der Dokumentgrammatik dient (0.4 für XCES, 4.1 in CES), sind die beiden ersteren nur dann sinnvoll nutzbar, wenn genau zwei Dokumente aligniert werden sollen, wobei `fromDoc` die Adresse der ersten und `toDoc` die der zweiten Instanz speichert (Ide, Priest-Dorman *et al.* 1996). In der XCES-Version sind beide Attribute vom Typ `xs:anyURI`, die CES-Variante nutzt hier den `CDATA`-Token-Typ.

Ein `linkGrp`-Element kann eine beliebige Anzahl `link`-Elemente beinhalten, die über `align`-Kindelemente auf die entsprechenden Zieldokumente (inkl. evtl. Ankerpunkt in der Datei) verweisen. Über optionale Attribute kann `linkGrp` weiter spezifiziert werden: Das `type`-Attribut gibt Auskunft über die mit dieser Gruppe assoziierten Komponenten der Dateien, die aligniert werden (z. B. Abschnitte), das `targType`-Attribut bezeichnet die konkreten Daten, die verknüpft werden (z. B. Absätze) und das Attribut `domains` kann genutzt werden, um Bezeichner der Elemente zu benennen, deren Kindelemente (oder Nachfahren) verknüpft werden.[14]

Die valide Instanz in Listing 7.8 verwendet das aktuelle XPOINTER XPOINTER() SCHEME zur Demonstration.

[14] Ide, Priest-Dorman *et al.* (1996) sind auch in diesem Punkt nicht ganz eindeutig bzgl. der Abgrenzung der Attribute `type` und `targType`. Zu ersterem heißt es: „indicates the type of data with which the group is associated, e.g., paragraph data, titles, etc.“, während `targType` definiert wird als „indicates the type of data being linked, e.g., paragraph, sentence, etc.“.

Listing 7.8: Beispiel-Instanz nach der cesAlign.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<cesAlign xmlns="http://www.xces.org/schema/2003"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.xces.org/schema/2003 xcesAlign.xsd"
 version="1">
 <linkList>
  <linkGrp type="Abschnitte" targType="paragraph" domains="p">
   <link>
    <align xlink:href="p.xml#xpointer(/cesDoc/text/body/p[1])" n="Primärdaten"/>
    <align xlink:href="a1.xml#xpointer(/cesAna/struct[1])" n="Annotation 1"/>
    <align xlink:href="a2.xml#xpointer(/cesAna/struct[1])" n="Annotation 2"/>
   </link>
  </linkGrp>
 </linkList>
</cesAlign>
```

Durch das xlink:href-Attribut der align-Elemente werden die einzelnen Dateien zueinander in Beziehung gesetzt. Im Beispiel-Listing wird der erste Absatz der anhand der cesDoc annotierten Primärdatendatei p.xml mit der ersten Merkmalsstruktur der beiden Annotationsdateien a1.xml und a2.xml verknüpft.

Das XML-Schema cesAna.xsd wirkt von einem technischen Standpunkt her nicht ganz ausgereift: Abgesehen von mehrfach verwendeten Attributen wie fromDoc und toDoc, die trotz gleicher Verwendung und identischen Aufbaus an verschiedenen Stellen im Dokument lokal deklariert sind, ist unterhalb des link-Elements ein Identitäts-Constraint per xs:unique eingeführt, der allerdings einen irritierenden Kommentar beinhaltet:

```
<xs:element name="link" type="xces:xlinkType" maxOccurs="unbounded">
 <xs:unique name="UniqueTargets">
  <xs:selector xpath="xces:align"/>
  <xs:field xpath="@n"/>
 </xs:unique>
 <!-- Each align element must have a unique n element. That is, each must refer to a
      different translation -->
</xs:element>
```

Abgesehen davon, dass es sich nicht um ein n-Element, sondern ein Attribut handelt, dass von dem XPATH-Ausdruck innerhalb des xpath-Attributs des xs:field-Elements selektiert wird, ist die weiterführende Begründung, dass ein align-Element sich eindeutig auf ein translation-Element beziehen muss, problematisch, da dieses Element Teil des optionalen Headers ist. Zumal es den Anschein hat, dass der Kommentar kopiert wurde, da das Element an dieser Stelle in der Dokumenthierarchie selbst bei genutztem Header nicht vorkommen könnte. Dazu kommt, dass das Attribut n in der einzig verfügbaren Dokumentation (Ide, Priest-Dorman *et al.* 1996) als global vorkommendes Attribut wie folgt definiert wird: „a number or other label for the element, not necessarily unique within the document.“ Was allerdings durch den Constraint erreicht wird, ist, dass der Wert des n-Attributs für alle align-Elemente unterhalb von link eindeutig sein muss (allerdings nicht dokumentweit eindeutig), wobei die Aussagekraft des Attributs – auch aufgrund des Namens – begrenzt bleibt. Andere Teile der Dokumentgrammatik wurden nur auskommentiert, was nicht für eine sorgfältige Schlussredaktion der Dateien spricht.

7.7 Überlappende Strukturen

Da CES auf das Konzept der Standoff-Annotation setzt, sind überlappende Annotationen prinzipiell unproblematisch (vgl. auch die Ausführungen in Ide 1998, Anhang 10, „Overlapping hierarchies“). Aus diesem Grund wird an dieser Stelle nicht näher darauf eingegangen.

7.8 Verbreitung und Einsatz

CES hat als De-facto-Standard eine recht hohe Verbreitung – mit einigen Einschränkungen, die im Folgenden näher erläutert werden.

Unter der URL `http://www.cs.vassar.edu/CES/CES-P.html` werden einige Projekte aufgeführt, die CES bzw. XCES einsetzen. Nur zu einigen wenigen sind noch verwertbare Informationen öffentlich zugänglich, darunter das bereits erwähnte „MULTEXT“ bzw. dessen Nebenprojekt „MULTEXT-East“ (Laufzeit von 1995 bis 1997).[15] Das von der Europäischen Kommission geförderte Projekt „Consortium for Central European Dictionary Encoding“ (CONCEDE, Laufzeit von 1998 bis 2000)[16] war an der Weiterentwicklung der SGML-DTD des Corpus Encoding Standard zur XML-Version beteiligt. Gegenstand des Projekts war die Entwicklung lexikalischer Ressourcen für die ost-europäischen Sprachen Bulgarisch, Tschechisch, Estnisch, Ungarisch, Rumänisch und Slowenisch. Das zu diesem Zweck entwickelte Annotationsformat zur Strukturierung lexikalischer Datenbanken wurde zunächst als SGML-DTD und später in Form einer XML-DTD definiert. Es ist dokumentiert in Kilgarriff (1999).[17]

Das von 1995 bis 2000 durchgeführte Projekt „ARCADE“, dessen Ziel die Weiterentwicklung von Werkzeugen zur Verarbeitung paralleler Korpora war, hat zu diesem Zweck ebenfalls CES-annotierte Korpusdaten erstellt.[18] Ein an der *Universität Pompeu Fabra* in Barcelona durchgeführtes plurilinguales Korpus-Projekt verwendet CES-Annotation als Grundlage für die Strukturierung der Primärdaten.[19]

Ausgehend vom zunächst durch das *Institut für deutsche Sprache* (*IDS*) in Mannheim koordinierten Projekt „Trans-European Language Resource Infrastructure“ (TELRI I, 1995–1998) wurde mit dem *TELRI Research Archive of Computational Tools and Resources* (TRACTOR) im Januar 2000 als Teil des Projekts TELRI II (1999-2001) an der *Universität Birmingham* (bis Dezember 2000 am *IDS*) eine internationale Ressource für Korpuslinguisten verschiedener europäischer Sprachen eingerichtet, die neben annotierten Korpora (inkl. paralleler Korpora) auch Software wie Perl-Skripte zur Verarbeitung annotierter Daten anbietet (Wynne 2001).[20] Ein Teil der dort verfügbaren

[15] Weitere Informationen zum Projekt finden sich unter der Adresse `http://nl.ijs.si/ME/`, zuletzt abgerufen am 19.04.2012.

[16] Die Projekt-Homepage mit weiteren Informationen unter der Adresse `http://www.itri.brighton.ac.uk/projects/concede/` einsehbar, zuletzt abgerufen am 19.04.2012.

[17] Die entsprechenden DTDs lassen sich über die Projekt-Homepage herunterladen.

[18] Weitere Informationen sind unter der URL `http://aune.lpl.univ-aix.fr/projects/arcade/index-en.html` einsehbar, zuletzt abgerufen am 19.04.2012.

[19] Der eindeutige Projektbezeichner ist unklar. Informationen sind weiterhin über die URL `http://www.iula.upf.edu/corpus/corpusuk.htm` einsehbar, zuletzt abgerufen am 19.04.2012.

[20] Nähere Informationen dazu unter `http://tractor.bham.ac.uk/tractor/`, zuletzt abgerufen am

Texte ist ebenfalls nach (X)CES (sowohl SGML als auch XML) annotiert. Jedoch muss angesichts des Alters davon ausgegangen werden, dass auch die XML-Annotation der ursprünglichen CES-Fassung der `cesAna.dtd` entspricht. Interessanterweise sind die Katalog-Informationen zu den Korpora (Erjavec und Váradi 2001) in DocBook annotiert (vgl. Abschnitt 4.3).

Im Rahmen des Projekts DEREKO I („DEutsches REferenzKOrpus“), das durch das *IDS*, dem *Seminar für Sprachwissenschaft* (*SfS*) der *Universität Tübingen* und dem *Institut für Maschinelle Sprachverarbeitung* (*IMS*) in Stuttgart in den Jahren 1999 bis 2002 durchgeführt wurde, wurde ein automatisiert annotierter Textkorpus der deutschen Sprache erstellt. Das Korpus ist mit einer angepassten Version von XCES annotiert (Ule 2002), wobei die Änderungen sich hauptsächlich auf Annotationseinheiten unterhalb der Satzebene konzentrieren:

> The DEREKO linguistic annotation scheme extends the Corpus Encoding Standard (CES) [...] below the sentence level to cover, e. g. the annotation of chunks and topological fields, or information for POS tag majority voting. It therefore complements the IDS extensions applying to sentences and larger units (Dipper, Kermes *et al.* 2002, S. 12).

Interessant ist der Fakt, dass die DEREKO-Annotation ursprünglich in Standoff-Form erfolgen sollte, dieser Plan aber aufgrund unzureichender Unterstützung von XML-Linking-Techniken nicht weiter verfolgt wurde (Dipper, Kermes *et al.* 2002, S. 13). Als Ausgangsbasis für ein Annotationsformat wurde CES verwendet. Diese Konvertierung wurde durch das *IDS* ab ungefähr 1997 durchgeführt (CES Level 1-konform, s. Ule 2002, S. 1). Im Jahr 2005 wurde die Migration nach XML begonnen. Bei beiden Anpassungen wurden kleinere Abweichungen vorgenommen, die sich größtenteils auf Wiederherstellung der Kompatibilität zur TEI beziehen und dokumentiert sind (Institut für Deutsche Sprache 2012b). Irritierend ist bei dieser Aufstellung allerdings, dass sich die Modifikationen auf eine „XCES Revision 4.3“ beziehen. Auf der XCES-Webseite sind die XML-Schemata-Dateien mit der Angabe „1.0.4“ versioniert. Das Dokumentgrammatik `xcesDoc.xsd` beziffert die Version in den Kommentaren als „XCES Revision 0.4“, allerdings findet sich im `version`-Attribut des `schema`-Elements ebenfalls die Angabe 1.0.4.[21] Die entsprechende DTD `xcesDoc.dtd` (ebenfalls von der XCES-Webseite bezogen), enthält die Angaben „Revision: 3.19“ und „1996/06/13“, daher ist nicht klar, auf welcher Basis die Änderungen beruhen. Für weitere Unklarheiten sorgt die in Institut für Deutsche Sprache (2012b) aufgeführte Entität `x.token`, die weder Bestandteil der XSD- noch der DTD-Version von XCES ist, die auf der XCES-Webseite zur Verfügung gestellt werden.

Weitere Versionen der XCES-DTDs befinden sich auf der Webseite des GATE-Projekts.[22]

19.04.2012, die Mitgliedschaft ist kostenpflichtig. Informationen zu den TELRI-Projekten finden sich unter der URL `http://telri.nytud.hu/`, zuletzt abgerufen am 19.04.2012.

[21] Diese Art der Versionierung ist zwar prinzipiell zulässig, hat aber den Nachteil, dass ein validierender Parser eine solche interne Angabe ignoriert und sie daher nicht zur Abweisung von Instanzen herangezogen werden kann, die eine andere Schema-Version nutzen.

[22] Unter der Adresse `http://gate.ac.uk/gate/src/gate/resources/texts/xces/dtd/` sind u. a. die DTDs `xcesAlign.dtd` und `xcesAna.dtd`, `xcesDoc.dtd` zum Download verfügbar, zuletzt abgerufen am 19.04.2012.

Diese werden als „Revision 345“ vom 13. Juni 2000 angegeben, allerdings fehlt auch hier besagte Entität `x.token`. Nach Anfrage beim *IDS* wurde bestätigt, dass die Angaben auf der Webseite ungenau sind. Ausgangspunkt war die SGML-CES-Version 4.3, die auch die besagte Entität enthält. Laut Institut für Deutsche Sprache (2012a) sind auch die anderen im System COSMAS II nutzbaren Korpora beim *IDS* seit April 2010 im XCES-Format kodiert. Aktuell ist dieses Annotationsformat als proprietär anzusehen, da es weder der TEI P5 noch der aktuellen XCES-Fassung entspricht (Kupietz, Belica *et al.* 2010, S. 1850). Problematisch ist in diesem Zusammenhang, dass zumindest die ursprüngliche SGML-basierte CES-Version durch die in der TEI vorgesehene Modifizierungsebene weiterhin TEI-konform (zur P3) bleibt, da sich die Änderungen sowohl in Form einer eingeebneten Dokumentgrammatik als auch als *TEI Extension* nutzen lassen. Die vom *IDS* durchgeführten zusätzlichen Änderungen sind dagegen nur in Form einer einzelnen Dokumentgrammatik (*flattened*) verfügbar. Es existieren allerdings Pläne, eine Konvertierung in ein TEI-konformes Tagset (nach P5) vorzunehmen, so dass auch aktuellere Entwicklungen der TEI eingesetzt werden können (Kupietz, Schonefeld *et al.* 2010, S. 40), mit dem Abschluss der Umstellung kann in 2012 gerechnet werden (Sperberg-McQueen 2011). Unklar bleibt allerdings, ob diese erweiterte und P5-konforme Version dann auch als offizielle XCES-Version veröffentlicht wird oder nur im *IDS* Verwendung findet.

Das 2008 erstmalig veröffentlichte „WordNet Gloss Disambiguation Project“ enthält 117,659 annotierte Glossen, die neben anderen Notationen auch in Form von XCES-Standoff-Instanzen verfügbar gemacht werden.[23] Neben den unannotierten Primärdaten sind jeweils eine Header-Datei, die Glossen-Struktur (in Form von `struct`- und `feat`-Elementen, vgl. Listing 7.7) sowie eine Token- und Sense Tag-Annotationsebene vorhanden. In der Dokumentation (in Form einer `README`-Datei) wird darauf hingewiesen, dass XCES in der aktuellen Form für die Annotation diskontinuierlicher Einheiten (bei Mehrwortformen, als Beispiel wird „personal or business relationship“ genannt) nicht vollständig geeignet ist, weshalb diese zusätzlich in einem alternativen Format ausgezeichnet wurden. Davon abgesehen sind die XCES-Instanzen valide gegenüber dem auf der XCES-Webseite vorgehaltenem XML-Schema `xcesAna.xsd`.

Auch die erste Veröffentlichung des „American National Corpus“ (ANC) erfolgte in Form von XCES-Instanzen.[24] Davon abgeleitet wurde der „Open American National Corpus“ (OANC). Mit 14.623.927 Wörtern, die neben einer Tokenisierung auch POS-Annotationen (vier unterschiedliche Tagsets) und Auszeichnungen von Nominal- und Verbal-Chunks enthalten (dazu kommen noch zusätzliche linguistische Annotationen, die für einzelne Teile des Korpus vorhanden sind), stellt er sicherlich eine der größten frei verfügbaren XCES-Anwendungen dar.[25] Der Vorschlag für das Vorhaben findet

[23] Download unter der URL `http://wordnetcode.princeton.edu/glosstag-files/`, zuletzt abgerufen am 19.04.2012.

[24] Weitere Angaben über das Tagset sind einsehbar unter `http://americannationalcorpus.org/FirstRelease/encoding.html`, zuletzt abgerufen am 19.04.2012.

[25] Quelle aller Angaben: `http://anc.org/` nebst Unterseiten, zuletzt abgerufen am 19.04.2012. Die Angaben zur Korpusgröße sind uneinheitlich: Während auf der Startseite `http://anc.org/` von „approximately 15 million words“ gesprochen wird, findet sich auf `http://anc.org/OANC/` die Angabe „over 14 million words“ sowie die vollständige Summe, unterteilt in 11.406.155 Wörter geschriebener und 3.217.772 Wörter gesprochener Sprache. Der vollständige, nicht offen verfügbare

sich bereits in Fillmore *et al.* (1998), eine erste Fassung des ANC wird in Ide und Suderman (2004) beschrieben. In seiner zweiten Version (Second Release) wurde der OANC zunächst XCES-kodiert zum freien Download zur Verfügung gestellt.[26] Diese Version enthält pro Korpuseintrag eine Textdatei mit den Primärdaten, eine Header-Datei mit der Endung `anc` und Annotationsebenen (in jeweils separaten Dateien) für die logische Dokumentstruktur, Satzgrenzen, Nominal- und Verbalchunks, sowie eine Part-of-Speech-Annotation (erzeugt durch den HeppleTagger, der Teil des GATE-Frameworks ist). Interessanterweise sind alle Annotationsdateien prinzipiell nur gegen die Dokumentgrammatik `xcesAna.xsd` validierbar, d. h., es gibt keine Instanzen gemäß der `xcesDoc.xsd`-Dokumentgrammatik, da die Primärdaten vollständig unannotiert sind.[27] Die Annotationen folgen dem in Listing 7.7 und in Suderman und Ide (2006) dargestellten Aufbau. Die aktuelle Fassung des OANC ist allerdings mittels GrAF rekodiert worden (vgl. Kapitel 9).

Habert *et al.* (2001) beschreiben die Nutzung von XCES als Basis eines Korpus medizinischer Texte. Den Angaben zufolge liegt das Hauptaugenmerk hier allerdings eher auf den Metadaten, so dass die Annotation der Texte nur Level 1-konform erfolgt. Hervorgehoben wird die Möglichkeit mittels XSLT Teilkorpora oder Zusammenfassungen zu extrahieren – diese steht aber prinzipiell bei allen XML-basierten Annotationsformaten zur Verfügung.

Bański (2001) diskutiert den Einsatz von XCES für das „IPI PAN Korpus“, das erste größere frei verfügbare Korpus polnischer Sprache (Przepiórkowski 2004). Besonderes Augenmerk gilt der Upconversion (bzw. Up-translation), also der schrittweisen Anreicherung von Annotationen im Zuge der weiteren Verarbeitung. Das „IPI PAN Korpus“ wird im Standoff-Verfahren annotiert:

> For the IPI PAN corpus, two kinds of stand-off annotation documents are initially planned. The first kind will contain sentence segmentation for the given text. It will refer to the base text via one-way links. The other kind of remote markup documents will contain morphosyntactic annotation. This kind of annotation will not reference the base text but rather the document with <s>-alignment annotation [...] (Bański 2001, S. 16).

Die 2004 veröffentlichte erste Version des „IPI PAN Korpus“ nutzt eine leicht modifizierte Version der XCES-Dokumentgrammatiken (Przepiórkowski 2004; Przepiórkowski, Krynicki *et al.* 2004), wobei jeweils eine Datei mit Metadaten (`header.xml`), eine Level 1-konforme Segmentierung der Primärdaten (`text.xml`) und eine morphosyntaktische Annotation (`morph.xml`) pro Korpuseintrag existieren. Die im März 2006 veröffentlichte zweite Version des Korpus enthält über 250 Millionen Segmente (zur Darstellung der

ANC besteht aus rund 22 Millionen Wörtern. Zielgröße des Korpus ist die Menge von 100 Millionen Wörtern.

[26] Diese Version lässt sich als Zip-Datei unter der Adresse `http://anc.org/OANC/OANC-1.0.1-UTF8.zip` herunterladen und wird auf der Seite `http://anc.org/OANC/index.html` unter dem Punkt „PREVIOUS VERSIONS“ als „Open ANC in the original XML format“ bezeichnet. Beide URLs zuletzt abgerufen am 19.04.2012.

[27] Die Einschränkung „prinzipiell“ resultiert aus der Tatsache, dass die Instanzen nicht valide gegenüber den XSD-Dateien der XCES-Webseite sind.

Segment-Grenzen sei verwiesen auf Przepiórkowski 2004). Die unter der *GNU General Public License* (GPL) stehende Version des „Frequency dictionary of contemporary Polish“ enthält dagegen nur jeweils Metadaten und morphosyntaktische Annotation, die sich anhand der ebenfalls zur Verfügung gestellten angepassten DTD `xcesAnaIPI.dtd` validieren lassen.[28]

Przepiórkowski, Górski *et al.* (2008); Bański und Przepiórkowski (2009) folgend, nutzt der „National Corpus of Polish“ (NKJP), der auf den Arbeiten des „IPI PAN Korpus“ sowie zwei weiteren Korpora aufbaut, nicht mehr XCES sondern eine zur P5 kompatible erweiterte Standoff-Annotation (vgl. Abschnitt 5.8) und verwendet ein minimales Inline-Markup der logischen Dokumentstruktur (innerhalb der Primärdaten) als Ausgangspunkt für die weiteren Annotationsebenen (Bański 2010). Wie Bański und Przepiórkowski (2010a) anmerken, ist dies ein Resultat der unzureichenden Dokumentation und verwirrenden Versionierung von XCES.[29]

7.9 Beurteilung

Die Beurteilung des CORPUS ENCODING STANDARD fällt heterogen aus: Als die SGML-Fassung entwickelt wurde, war sie als TEI-konforme Erweiterung der P3 für die Annotation linguistischer Korpora sehr sinnvoll. Vor allem die Möglichkeit der schrittweisen Ergänzung der Annotation brachte einen neuen Aspekt in die Korpusentwicklung mit:

> As already mentioned, the CES is in fact a set of standards, ranging from the simplest type of annotation, achieved in most cases almost cost-freely, to complex annotation that needs human intervention on the one hand, but on the other, it is very well suited for NLP applications. Hence, it is possible to start off small and cheap at the beginning, and – in time – to refine the annotation, when more financial resources and manpower are available. What is important is that the path of this so-called up-translation is already laid out by the CES guidelines. (Bański 2001, S. 5)

Mit der fortschreitenden Entwicklung der TEI haben sowohl CES als auch XCES allerdings den Anschluss verloren. Die seit einigen Jahren bestehende Diskrepanz zwischen der aktuellen Version der TEI GUIDELINES und der XCES-Dokumentgrammatik führt dazu, dass XCES keine TEI-Anwendung mehr ist (was allerdings zumindest für die vom *IDS* entwickelte Fassung in naher Zukunft wieder geändert wird, vgl. Abschnitt 7.8). Demgegenüber steht die starke Verbreitung der Spezifikation. Allerdings ist nicht zu übersehen, dass der Anspruch der Nutzer und die offiziell zur Verfügung gestellte Dokumentgrammatik nicht deckungsgleich sind: Abgesehen davon, dass alle größeren Projekte Modifikationen an den Dokumentgrammatiken vorgenommen haben, ist eine Tendenz zurück zu einer TEI-kompatiblen Version vernehmbar, die durch das mit der P5 eingeführte Instrument ODD deutlich einfacher als zuvor ist. Es bleibt abzuwarten,

[28] Alle Downloads befinden sich unter `http://korpus.pl/index.php?page=download`, zuletzt abgerufen am 19.04.2012.

[29] Weitere Informationen zum NCP finden sich unter `http://nkjp.pl/`, zuletzt abgerufen am 19.04.2012.

ob und wann die auf der XCES-Webseite in Aussicht gestellte Anpassung des XCES an die P5 für alle nutzbar erfolgt.[30] Zusätzlich kann davon ausgegangen werden, dass XCES aktuell durch die abschließenden Arbeiten an LINGUISTIC ANNOTATION FRAMEWORK (vgl. Kapitel 9) weiterhin an Verbreitung einbüßen wird. Dies lässt sich gerade am Beispiel des OANC (erste vs. zweite Fassung) eindrucksvoll beobachten. Durch den Umstand, dass die Metadaten und das generelle Konzept der Standoff-Annotation erhalten bleiben, dürfte eine Transformation zwischen den beiden Formaten allerdings prinzipiell möglich sein.[31]

Sehr problematisch ist allerdings der Umstand, dass die Dokumentation allenfalls als rudimentär zu bezeichnen ist. Die mehrfach als Handbuch sowohl für CES als auch für XCES herangezogenen Ausführungen in Ide, Priest-Dorman *et al.* (1996) sind unvollständig und können im Fall von XCES aufgrund der diskutierten Abweichungen der Spezifikationen nicht genutzt werden. Dies ist besonders kritisch anzumerken, da CES laut EAGLES-Webseite im Status „R/F“ (*Recommendations/Formal specifications and explicit guidelines*) eingestuft wird.[32] Auch bestehen Unklarheiten bzgl. der aktuellen Version von XCES, was die Versionierung von Korpora unnötig erschwert (vgl. die unklaren Versionsangaben in Bezug auf die durch das *IDS* genutzte Fassung).

Aus Sicht der verwendeten Constraint Language ist die Verwendung von XSD als Grammatikformalismus aus technischen Gründen gerechtfertigt, da einige Merkmale verwendet werden, die in DTD nicht vorhanden sind (beispielsweise die Verwendung von `xs:unique` in `xcesAlign.xsd` oder einiger Datentypen). Abgesehen davon ist die formale Ausdrucksstärke als LTG einzustufen, da es keine Hinweise auf konkurrierende Nichtterminale, d. h., Elemente gleichen Namens mit unterschiedlichem Inhaltsmodell je nach Kontext, gibt.[33] Allerdings waren die XSD-Dateien über einen längeren Zeitraum nicht valide, da die zur Bearbeitung herangezogene Software (ALTOVA XMLSPY) in der damaligen Version ungenau gegenüber der Spezifikation von XML SCHEMA (spezifisch: Thompson *et al.* 2004) war. Auf Initiative des Verfassers (im Rahmen eines Treffens mit Nancy Ide und Keith Sudermann auf der LREC 2008) wurden diese Fehler korrigiert.

Insgesamt bleibt also ein gemischtes Fazit, dem sich auch andere Autoren anschließen. Hier sei wieder auf Bański und Przepiórkowski (2010a, S. 98) verwiesen: „Taking also into account the lack of documentation and the potential confusion concerning its versioning, XCES turns out to be unsuitable for the purposes of NKJP.“

[30] Zitat: „XCES is continually under development and future work will include making the XCES compliant with TEI P5.“. Quelle: `http://www.xces.org/`, zuletzt abgerufen am 19.04.2012.

[31] Es ist davon auszugehen, dass eine solche Transformation für den Übergang der beiden OANC-Versionen eingesetzt wurde.

[32] Zusätzlich entspricht die Angabe „Edited for Release = YES“ einem „final state for public release“.

[33] In der Datei `anc-xces.xsd`, die ebenfalls zu den auf der XCES-Webseite herunterladbaren XML-Schemata gehört, finden sich den benannten Gruppen `title.group`, `author.group` und `item.group` jeweils zwei Elemente, die sich durch ein Präfix „h.“ und den Datentyp (einmal ein aus der TEI abgeleiter Datentyp, einmal ein originärer XCES-Datentyp) unterscheiden, beispielsweise `h.title` und `title`. Hier wäre die formale Ausdrucksstärke einer RTG und damit der Einsatz von RELAX NG notwendig, um diese Inhaltsmodelle mit gleichen Genneric Iddentifier modellieren zu können. Die XSD-Dateien stehen unter der URL `http://www.xces.org/schema/xml-ces/xml-ces.zip` zum Download, zuletzt abgerufen am 19.04.2012.

8

Data Category Registry for Language Resources

Anmerkung Dieses Kapitel stellt nicht alle Bestandteile der Norm ISO 12620:2009 vor, sondern nur die für den aktuellen Untersuchungsgegenstand der Arbeit relevanten Teile. Außer Acht gelassen werden die organisatorischen Prozesse zur Einreichung bzw. Überarbeitung bestehender Datenkategorien.

Ein bereits mehrfach angesprochenes Problem bei der Annotation sprachlicher Informationen ist – neben der Auswahl der Granularität der Auszeichnung – das der Benennung der Komponenten des Annotationsinventars, also der spezifischen Elemente und Attribute. Diese ist zum einen abhängig von der linguistischen Betrachtungsebene (im Sinne einer konzeptuellen Ebene, vgl. Abschnitt 3.5) aber auch vom Untersuchungsgegenstand (so enthalten nicht alle Sprachen die gleichen morphosyntaktischen Merkmale) und der Sprache, in der die Benennung der Element- und Attributnamen erfolgt (also die Zeichenkette, die den Generic Identifier bildet). Da eine gesicherte Erkenntnis der letzten Dekaden die ist, dass man sich nie auf *das eine* Auszeichnungsinventar (sowohl in Bezug auf die Anzahl, die Struktur als auch die Namen der Elemente und Attribute) wird einigen können, gewinnen zwei Strategien mehr und mehr an Bedeutung: die Bereitstellung eines generischen Annotationsmechanismus' (z. B. in Form der in Kapitel 6 diskutierten Merkmalsstrukturen) und die Schaffung einer zentralen Kategorisierungsstelle zur standardisierten Verwaltung von Annotationsmerkmalen (also den Merkmalen und deren konkreten Werten). Eine solche wird im allgemeinen als *Data Category Registry* (DCR, etwa „Datenkategorienregistrierung") bezeichnet. Eine DCR ist ein Verzeichnis von Datenkategorien (*Data Category*, DC), die jeweils für sich Resultat von Standardisierungsbemühungen im Bereich einer gewissen Disziplin bzw. eines genau definierbaren Unterbereichs einer Disziplin sind. Ide und Romary (2004a, S. 135) definieren eine DC als einen „elementary descriptor used in a linguistic annotation scheme". Anders ausgedrückt sind Datenkategorien Domänen-spezifische Konzepte – im vorliegenden Fall Konzepte der Domäne linguistischer Daten. Einfache Beispiele für

Datenkategorien sind „Grammatisches Geschlecht“ oder auch „Akkusativ“. Als Beispiel dienen die beiden XML-Annotationen aus Listing 8.1 und 8.2.

Listing 8.1: Annotation eines Absatzes (A)

```
<text>
<para>Ein Absatz</para>
</text>
```

Listing 8.2: Annotation eines Absatzes (B)

```
<text>
<p>Ein Absatz</p>
</text>
```

Die Elemente `para` und `p` unterscheiden sich in ihrem Generic Identifier, bezeichnen aber beide das Konzept eines Absatzes in der logischen Dokumentstruktur. Was für einen menschlichen Leser trivial erscheinen mag (die Abbildung beider Elementnamen auf ein und dasselbe Konzept) ist für eine maschinelle Verarbeitung sehr schwer. Verweisen jedoch beide Elementnamen auf die Datenkategorie „Absatz in der logischen Dokumentstruktur“ (z. B. repräsentiert durch einen URI) ist eine gegenseitige Abbildung problemlos möglich. Damit ergänzt eine solche DCR die Syntax einer Auszeichnungssprache um eine semantische Komponente und adressiert damit einen wesentlichen Aspekt einer gemeinsamen Sprache: „Real interoperability is a function of shared semantics, not syntax.“ (Bray 2005, S. 3) An dieser Stelle sei nur darauf verwiesen, dass es auch Vertreter der Position gibt, eine Dokumentgrammatik für Auszeichnungssprachen würde eine Semantik spezifizieren. Die vorherrschende Meinung sieht XML Schema und andere Constraint Languages allerdings nur als Mittel zum Ausdruck syntaktischer Beschränkungen an.[1] Ebenso argumentieren Wilde und Glushko (2008, S. 43), wenn sie wie folgt ausführen:

> [...] XML has almost no predefined semantics (the only exception being one predefined attribute for identifying languages) [...] XML's ability to define names for elements and attributes, and the widespread assumption that these names have some intrinsic semantics, often cause victims to assume that the semantics of an XML document are self-evident, openly available just by looking at it and understanding the names.

Ein weiteres Einsatzgebiet einer DCR neben der Benennung von Element- und Attributnamen ist deren Einsatz in Metadaten. Da diese immens wichtig sind, um in größeren Korpora einzelne (oder bestimmte) Einträge zu finden, können standardisierte Datenkategorien hier für eine nachhaltigere Nutzung sorgen:

> The use of uniform data category names and definitions within the same resource domain (e. g., among terminological, lexicographical, text corpus, etc. resources), at least at the interchange level, contributes to system coherence and enhances the re-usability of data. (Ide und Romary 2004a, S. 135).

[1] Eine diesbezügliche Diskussion auf der Mailingliste xmlschema-dev kann unter der URL `http://lists.w3.org/Archives/Public/xmlschema-dev/2011Nov/0018.html` nachgelesen werden (zuletzt abgerufen am 19.04.2012). Die Ergebnisse sind in Costello (2011) zusammengefasst.

Darüber hinaus können nicht nur die Kategorien selbst, sondern auch die Verfahren zur Definition neuer Einträge angeglichen werden, um Interoperabilität zu fördern (Ide und Romary 2004a, S. 135).

Für Sprachressourcen wurde mit der internationalen Norm ISO 12620:2009 eine solche DCR als Ersatz für ISO 12620:1999 verabschiedet. ISO 12620:2009 beschreibt dabei ein Datenmodell für eine Datenkategorienregistrierung und die möglichen Verfahren, um Daten in einer DCR zu manipulieren, sowie das Austauschformat Data Category Interchange Format (DCIF). Während der Entwicklung wurde die Spezifikation ursprünglich in zwei Teilen (ISO/CD 12620-1 (N 488); ISO/CD 12620-2) entwickelt, beide sind noch im Status eines Committee Draft öffentlich zugänglich.[2]

Bestand die ursprüngliche Version der Norm ISO 12620:1999 noch aus einer festen Liste von Kategorien, die auf Basis verschiedener Standards aus ISO/TC 37 hervorgegangen sind, bietet ISO 12620:2009 die Möglichkeit, flexibel auf neue Anforderungen reagieren zu können. Dazu gehört auch die Möglichkeit, eine vorhandene Registrierung (in Form einer Implementation, vgl. Abschnitt 8.2) kollektiv zu nutzen – entweder, um auf Datenkategorien lesend zuzugreifen oder neue zu ergänzen (Broeder, Kemps-Snijders *et al.* 2010, S. 44). Dabei ist ISO 12620:2009 kompatibel zu einer Familie von ISO/IEC Normen aus dem Bereich der Metadaten, die aktuell aus sechs Spezifikationen besteht (ISO/IEC 11179-3:2003; ISO/IEC 11179-1:2004; ISO/IEC 11179-6:2004; ISO/IEC 11179-4:2004; ISO/IEC 11179-5:2005; ISO/IEC 11179-2:2005), wobei zu beachten ist, dass eine DC im Sinne von ISO 12620:2009 einem *Datenelement* (*Data Element*) aus den genannten Spezifikationen entspricht, die Verwendung von Termini also nicht einheitlich ist: „In ISO/IEC 11179 the basic container for data is called a data element. It may exist purely as an abstraction or exist in some application system.“ (ISO/IEC 11179-1:2004, S. vi).

Innerhalb einer DCR können unterschiedliche Kategorieselektionen (*Data Category Selections*, DCS, teilweise auch als *Data Category Sets* bezeichnet) nebeneinander bestehen (wobei Überschneidungen zulässig sind). Diese bilden dann beispielsweise eine Teildomäne innerhalb der DCR ab.

8.1 Aufbau

Eine Datenkategorie im Sinne von ISO 12620:2009 besteht aus drei Bereichen: den administrativen Informationen (in ISO/CD 12620-1 (N 488), S. 9 *Administration Identification Level* genannt), den beschreibenden Informationen (*Description Level*) und den linguistischen (bzw. sprachbezogenen) Informationen. Die dem Verfasser vorliegenden, letzten öffentlich verfügbaren Versionen, ISO/CD 12620-1; ISO/CD 12620-2, sehen die linguistischen Informationen als Teil der beschreibenden Informationen an. Allerdings beziehen sich die aktuelleren Ausführungen zum Datenmodell auf der ISOcat-Webseite[3] (vgl. Abschnitt 8.2) explizit auf drei Bereiche, was auch durch die folgende Aussage untermauert wird: „One of the main changes to the previous draft of the standard is the revision of the DCR data model.“ Diese Struktur spiegelt sich auch im DCIF-Austauschsformat

[2] Der erste Teil ist zusätzlich in der etwas jüngeren Version ISO/CD 12620-1 verfügbar.
[3] Zu finden unter der URL `http://www.isocat.org/files/12620.html`.

wider (vgl. Listing 8.4). Ein Überblick über das DCR-Datenmodell ist der Abbildung 8.1 zu entnehmen.[4]

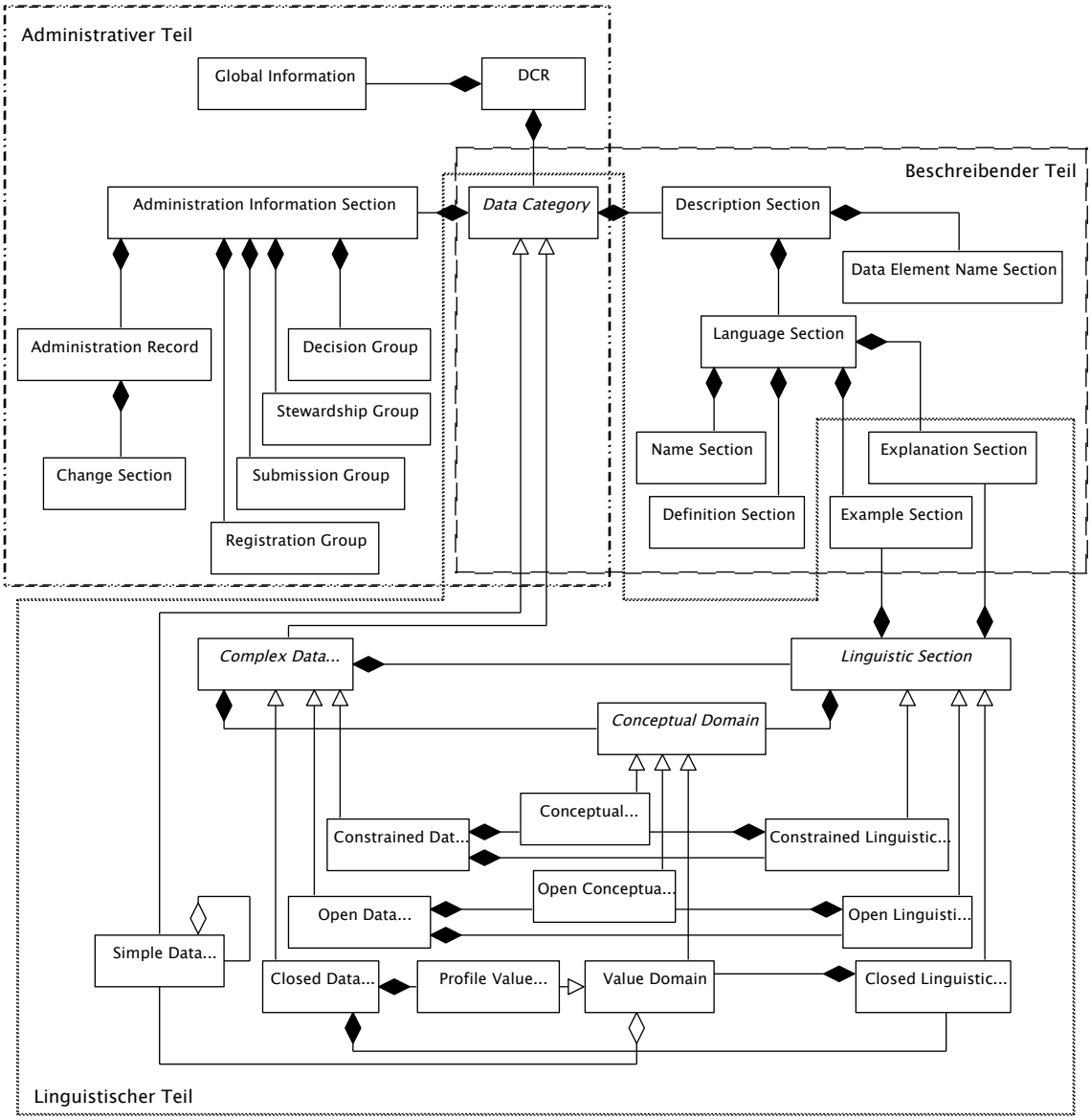


Abbildung 8.1: Das DCR-Datenmodell in der Übersicht

Sichtbar ist, dass eine DCR aus globalen Informationen und den Datenkategorien besteht. Der administrative Teil betrifft sowohl die DCR als Ganzes als auch die DC, während die beiden anderen Teile nur die Datenkategorien betreffen. Ein konkretes Beispiel aus der ISOCAT-Implementierung (vgl. Abschnitt 8.2) ist in Listing 8.4 dargestellt.

[4] Die ursprüngliche Grafik wurde der ISOCAT-Webseite von der Adresse `http://www.isocat.org/12620/model/DCR_data_model.svg` entnommen, zuletzt abgerufen am 19.04.2012.

Das DATA CATEGORY INTERCHANGE FORMAT (DCIF) wird hier nur in Kürze beschrieben, da die Notation aufgrund der vorhandenen Referenzimplementation ISOCAT von geringerer Relevanz als bei anderen Standards ist.[5] In ISO/CD 12620-1 (N 488) ist in Anhang A (also im informativen Teil der Spezifikation) eine DTD für das GENERIC MAPPING TOOL (GMT) angegeben, das in Ide, Kilgarriff *et al.* (2000); Ide und Romary (2001b) zusammen mit einem formalen Modell skizziert wird und auch als format-unabhängige Notation Teil der Norm TERMINOLOGICAL MARKUP FRAMEWORK (TMF, ISO 16642:2003) ist (vgl. auch die Abschnitte 4.2 und 7.5 dieser Arbeit).[6] Dessen Anwendung im Kontext syntaktischer Annotationen wird demonstriert in Ide und Romary (2001a), während Romary (2001) die Verwendung zur Repräsentation von Terminologie-Netzen diskutiert. Ide und Romary (2001b) schließlich nehmen eine Einordnung als Ausgangspunkt für neue Standardisierungsbemühungen im Rahmen von ISO/TC 37/SC 4 vor. Die in ISO/CD 12620-1 (N 488) gezeigte Version weicht von der in ISO 16642:2003 enthaltenen durch folgende Unterschiede ab: Für das Attribut `type` des `struct`-Elements sind andere Werte zulässig. Zusätzlich sind die beiden optionalen Attribute `id` und `target` dem Element hinzugefügt worden. Das Element `brack` kann – aufgrund geänderter Okkurrenzindikatoren – auch nur aus einem `brack`-Kindelement bestehen, während in ISO 16642:2003 mindestens ein `feat`-Element vorhanden sein muss. Das diesem Element zugeordnete Attribut `type` wurde entfernt, dafür wurde der Status des gleichnamigen Attributs des `annot`-Elements von *implied* (optional) auf *required* (obligatorisch) geändert.

Listing 8.3 zeigt die in ISO/CD 12620-1 (N 488), S. 22 aufgeführte Beispielinstanz zur Darstellung der beschreibenden Informationen (deutlich gemacht durch den Wert des `type`-Attributs des `struct`-Elements).

Listing 8.3: Beispielinstanz für das DCIF-Format

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <struct type="Desc">
3   <feat type="entry identifier">Gender</feat>
4   <feat type="profile">morpho−syntax</feat>
5   <brack>
6    <feat type="definition" xml:lang="fr">Catégorie reposant, selon les langues et les systèmes, sur la
         distinction naturelle entre les sexes ou sur des critères formels. Genre naturel, grammatical; genre animé,
         inanimé, genre féminin, masculin, neutre; genre adjectif, substantif des deux genres.</feat>
7    <feat type="source">www.atilf.inalf.fr Tlfi, GENRE, d, gramm</feat>
8   </brack>
9   <feat type="example">…</feat>
10  <feat type="explanation">…</feat>
11  <feat type="conceptual domain" target="#MASCULINE #FEMININE # NEUTER"/>
12  <struct type="LS">
13   <feat type="language">English</feat>
14   <brack>
15    <feat type="definition">…</feat>
16    <feat type="source">…</feat>
```

[5] Das Austauschformat ist dann relevant, sofern lokale DCRs eingerichtet werden sollen. In diesem Fall wäre ein Export aus einer vorhandenen in eine neu einzurichtende Installation im DCIF der korrekte Weg der Datenmigration. Da ISOCAT verschiedene Import- und Exportformate anbietet und auch die Eingabe neuer Datenkategorien über die Weboberfläche komfortabel möglich ist, ist davon auszugehen, dass DCIF nur selten zur Anwendung kommen wird.

[6] Eine entsprechende DTD findet sich unter `http://www.loria.fr/projets/TMF/DOC/DTD/tmf.dtd.txt`, zuletzt abgerufen am 19.04.2012. Zu beachten ist die Dateiendung „txt“.

```
  </brack>
  <feat type="example">…</feat>
  <feat type="explanation">…</feat>
  <struct type="NS">
   <feat type="name">Absolutive</feat>
  </struct>
 </struct>
</struct>
```

Es ist allerdings davon auszugehen, dass in der als Internationaler Standard ISO 12620:2009 verabschiedeten Fassung eine andere Struktur für das Austauschformat enthalten ist, da die von der Referenzimplementation ISOcat als DCIF-Export zur Verfügung gestellten Dateien erheblich davon abweichen, wie Listing 8.4 zeigt.

Listing 8.4: Aktueller DCIF-Export aus ISOcat

```
<?xml version="1.0" encoding="utf-8"?>
<dcif:dataCategorySelection xmlns:dcif="http://www.isocat.org/ns/dcif" dcif-version="1.2">
 <dcif:globalInformation>Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands</
     dcif:globalInformation>
 <dcif:dataCategory pid="http://www.isocat.org/datcat/DC-1297" type="complex">
  <dcif:administrationInformationSection>
   <dcif:administrationRecord>
    <dcif:identifier>grammaticalGender</dcif:identifier>
    <dcif:version>1:0</dcif:version>
    <dcif:registrationStatus>private</dcif:registrationStatus>
    <dcif:justification>Used in Morphosyntax, Terminology, Lexicography</dcif:justification>
    <dcif:origin>Morphosyntax; ISO 12620:1999</dcif:origin>
    <!-- [...] -->
   </dcif:administrationRecord>
  </dcif:administrationInformationSection>
  <dcif:descriptionSection>
   <dcif:profile>Morphosyntax</dcif:profile>
   <dcif:profile>Terminology</dcif:profile>
   <dcif:dataElementNameSection>
    <dcif:dataElementName>grammatical gender</dcif:dataElementName>
    <dcif:source>ISO 1262:1999</dcif:source>
   </dcif:dataElementNameSection>
   <dcif:languageSection>
    <dcif:language>en</dcif:language>
    <dcif:nameSection>
     <dcif:name xml:lang="en">grammatical gender</dcif:name>
     <dcif:nameStatus>standardized name</dcif:nameStatus>
    </dcif:nameSection>
    <dcif:definitionSection>
     <dcif:definition xml:lang="en">Category based on (depending on languages) the natural distinction
         between sex and formal criteria.</dcif:definition>
     <dcif:source>GP</dcif:source>
     <dcif:note xml:lang="en"/>
    </dcif:definitionSection>
    <dcif:definitionSection>
     <!-- [...] -->
    </dcif:definitionSection>
   </dcif:languageSection>
   <!-- [...] -->
  </dcif:descriptionSection>
  <dcif:conceptualDomain type="closed">
   <dcif:dataType>string</dcif:dataType>
   <dcif:profile>Morphosyntax</dcif:profile>
   <dcif:value pid="http://www.isocat.org/datcat/DC-1558"/>
   <dcif:value pid="http://www.isocat.org/datcat/DC-1880"/>
   <dcif:value pid="http://www.isocat.org/datcat/DC-1883"/>
   <dcif:value pid="http://www.isocat.org/datcat/DC-1884"/>
```

```
46    </dcif:conceptualDomain>
47    <!-- [...] -->
48    <dcif:linguisticSection type="closed">
49     <dcif:language>en</dcif:language>
50     <dcif:conceptualDomain type="closed">
51      <dcif:dataType>string</dcif:dataType>
52      <dcif:value pid="http://www.isocat.org/datcat/DC-1880"/>
53      <dcif:value pid="http://www.isocat.org/datcat/DC-1883"/>
54      <dcif:value pid="http://www.isocat.org/datcat/DC-1884"/>
55     </dcif:conceptualDomain>
56    </dcif:linguisticSection>
57    <!-- [...] -->
58   </dcif:dataCategory>
59  </dcif:dataCategorySelection>
```

Da diese Dateien sich auch gegen die auf der ISOCAT-Webseite bereitgestellte Dokumentgrammatik validieren lassen, können sie als korrekt angesehen werden.[7]

Listing 8.4 stellt die Datenkategorie „grammaticalGender“ (grammatikalisches Geschlecht) dar.[8] Dieser Eintrag wurde aus der ISOCAT-Implementation übernommen, da er eine große Nähe zur Beispiel-Datenkategorie aus Listing 8.3 aufweist, und somit die Unterschiede zwischen beiden Formaten deutlich macht. Die Benennung der einzelnen Abschnitte durch Elementnamen ist präziser als die Verwendung der Werte des `type`-Attributes in der ursprünglichen Fassung. Zu beachten ist, dass der Eintrag nicht nur aus den beschreibenden Informationen besteht, sondern aus einer vollständigen DCS. Der relevante Teil beginnt in Zeile 4 mit dem Element `dataCategory`. Es trägt einen eindeutigen Bezeichner (*Persistent Identifier*, Attribut `pid`), der in Form eines URI aufgebaut ist, was die spätere Referenzierung erlaubt (z. B. bei der Nutzung anderer Datenkategorien als Werte wie in Zeile 42). Kindelemente sind `administrationInformationSection`, `descriptionSection`, `conceptualDomain` sowie `linguisticSection`.[9] Über das `type`-Attribut kann eine Datenkategorie weiter spezifiziert werden. Offene DC (in diesem Fall ist der Wert des `type`-Attributs *open*) können beliebige Zeichenketten als Wert haben. Geschlossene (*closed*) Datenkategorien besitzen vordefinierte Werte (in Form einer Auswahlliste). Eingeschränkte (*constrained*) Kategorien sind ebenfalls nicht offen, allerdings besteht die Einschränkung der zulässigen Werte nicht in einer Auswahlliste, sondern in einem Schema-spezifischen Ausdruck (z. B. einem regulären Ausdruck). Diese drei Varianten sind allesamt komplexe Datenkategorien, d. h., sie bestehen aus der Kategorie und deren Werten. Demgegenüber *ist* eine einfache DC ein Wert (ein Beispiel hierfür ist die Referenz in Zeile 43 aus Listing 8.4). Container-Datenkategorien bestehen aus einer DC, die weitere komplexe Kategorien enthält. Diese Variante ist lt. ISOCAT-Webseite erst mit der Version 1.2 im Januar 2011 hinzugefügt worden.

Der administrative Teil (genauer: das Element `administrationRecord`) enthält

[7] Die Dokummentgrammatik lässt sich als RELAX-NG-Schema sowohl in der Compact Syntax als auch in der XML-Syntax unter der Adresse `http://www.isocat.org/files/12620.html` beziehen, zuletzt abgerufen am 19.04.2012.

[8] Die Bezeichner sollten immer in „Camel Case“-Schreibweise und auf Englisch formuliert werden – im Fließtext können Weißraum-Zeichen genutzt werden (Francopoulo, Declerck, Sornlertlamvanich *et al.* 2008, S. 3).

[9] Der URI setzt sich zusammen aus der ISOCAT-URL, dem Verzeichniseintrag „datcat“ und anschließend einem eindeutigen Schlüssel, vorangestellt durch die Zeichenkette „DC-“. Der eindeutige Schlüssel ist auch in der grafischen Darstellung in Abbildung 8.2 sichtbar.

einen administrativen Bezeichner (Element `identifier`), eine Versionsangabe des Eintrags[10], Angaben über den Status[11], Grund bzw. Motivation der Aufnahme dieser Datenkategorie in die DCR (`justification`) und zur Herkunft (`origin`). Optional sind auch noch erläuternde Anmerkungen (`explanatoryComment`), enthaltene Probleme (`unresolvedIssue`) sowie Datumsangaben (für den Zeitraum der Gültigkeit, `effectiveDate` bzw. `untilDate` sowie den Zeitpunkt der Erstellung oder Modifikation) im administrativen Teil erlaubt.

Der beschreibende Abschnitt enthält zunächst mindestens ein `profile`-Elemente zur Festlegung eines Profils, dass bei der Beschreibung der konzeptuellen Domäne referenziert werden kann. Es folgen die Elemente `dataElementNameSection` und `languageSection`, die in beliebiger Reihenfolge angeordnet werden können. In der `languageSection` (ein Element pro Sprache) werden natürlichsprachliche Informationen zur Datenkategorie gespeichert, hierzu kann auch ein `definitionSection`-Kindelement genutzt werden, das neben einer Definition weitere optionale Anmerkungen enthalten kann. Je mehr (sinnvolle) Informationen unterhalb von `languageSection`-Kindelementen gespeichert sind, umso einfacher wird für Benutzer die Zuordnung der Datenkategorie zum dahinter stehenden Konzept.

Die konzeptuelle Domäne, die mehrfach auftreten und über den Typ (*open*, *constrained* oder *closed* sind mögliche Attributwerte) weiter beschrieben werden kann, verweist – in Abhängigkeit mit dem zuvor definierten Profil auf andere Konzepte (sprich: Datenkategorien), die in Relation zur aktuellen DC stehen (über die Art der Relation wird keine Aussage gemacht). In Listing 8.4 werden beispielsweise ab Zeile 41 die Relationen zur konzeptuellen Domäne der Morphosyntax aufgeführt. Die vier `value`-Elemente referenzieren über das `pid`-Attribut die Einträge zu den Datenkategorien „commonGender“ (Zeile 42), „feminine“ (Zeile 43), „masculine“ (Zeile 44), und „neuter“ (Zeile 45). Diese letzten drei werden auch in der konzeptuellen Domäne des linguistischen Teils (Element `linguisticSection`) wieder referenziert. Hier können auch Beispiele (`exampleSection`), Erklärungen (`explanationSection`) oder allgemeine Anmerkungen (`note`) untergebracht werden.

Eine menschenlesbare Darstellung dieser Datenkategorie – entnommen aus der DCR-Implementierung ISOcat – ist in Abbildung 8.2 einsehbar.

[10] Laut einer auf dem am 10. Mai 2011 in Utrecht durchgeführten CLARIN-NL ISOcat-Workshop gehaltenen Präsentation, die unter der URL `http://www.clarin.nl/system/files/ISOcat-DC-specifications.pdf` einsehbar ist (zuletzt abgerufen am 19.04.2012), basiert das Versionierungssystem von ISOcat auf „Branching“, d. h., von der Version 1.0 gehen sowohl kleinere Änderungen – solche, die in einer höheren Nachkommastelle resultieren – als auch größere – und möglicherweise nicht-kompatible – Änderungen mit höherer Vorkommastelle aus.

[11] Das `registrationStatus`-Element kann die Werte *private*, *candidate*, *standard*, *deprecated* oder *superseded* annehmen, um den Stand der Registrierungsbemühungen abzubilden. Sinnvollerweise werden neue Einträge mit dem Status *private* versehen, der auch weniger obligatorische Angaben erfordert. Nach und nach kann eine Datenkategorie dann zum Wert *standard* erweitert werden – was allerdings auch die Zustimmung der organisatorischen Gremien hinter der DCR voraussetzt.

8.2 Implementierung und Einsatz

Mit ISOcat („Data Category Registry for ISO TC 37“) steht eine DCR für Sprachressourcen zur Verfügung, die vom *Max Planck Institute für Psycholinguistik* in Nijmegen betreut wird (formal korrekt ausgedrückt ist das *MPI* die *Registration Authority*). ISOcat stellt dabei die Referenzimplementation für ISO 12620:2009 dar. Die Plattform ist Web-basiert, und kann seit dem Sommer 2009 allgemein genutzt werden (zuvor war in einer Testphase der Zugang auf Experten beschränkt). Abbildung 8.2 zeigt die grafische Benutzeroberfläche nach erfolgter Anmeldung (eine Nutzung als Gast ist ebenfalls möglich) und Auswahl einer Datenkategorie (in diesem Fall zeigt die Abbildung den bereits bekannten Eintrag „grammaticalGender“).

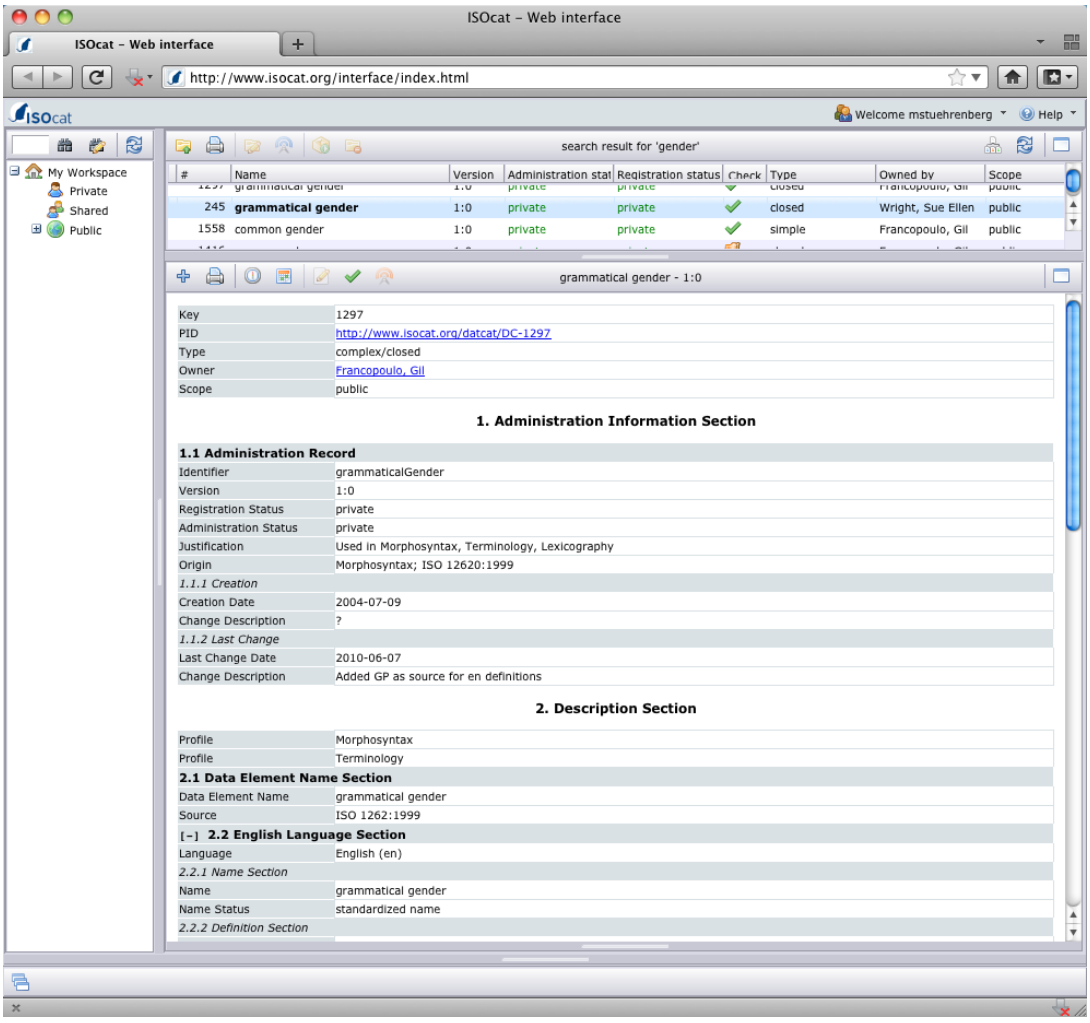


Abbildung 8.2: Grafische Oberfläche der ISOcat-Webanwendung (Screenshot)

Die Web-basierte Oberfläche erlaubt den Zugriff mit gängigen Browsern (auch wenn teilweise noch Probleme mit bestimmten Browserversionen existieren, wie die Hilfe-

seiten attestieren). Neben der Auflistung der Datenkategorien, die nach Themen oder Nutzergruppen im linken Bereich der Darstellung erfolgt, werden im rechten Teil oben die dieser Auswahl (DCS) zugehörigen DCs angezeigt, während darunter die Detailansicht des ausgewählten Eintrags erfolgt. Hier sind – abhängig von den Rechten des angemeldeten Nutzers – Änderungen und Ergänzungen der Datenkategorie möglich.

Nicht sichtbar in der Abbildung ist ein Fensterbereich unterhalb, der genutzt werden kann, um eine eigene DCS zu erstellen und diese im Anschluss zu exportieren. Aktuell stehen als Exportformat das im letzten Abschnitt diskutierte DCIF oder RDF zur Auswahl (die beiden weiteren Alternativen, RNG und XSD, sind nur teilweise funktionsfähig).

Durch eine Reihe von Projekten wurde und wird ISOcat mit Kategorien aus dem Bereich der Sprachressourcen angereichert, darunter gehören auch die Konzepte der GOLD-Ontology (Farrar und Langendoen 2003), die im Rahmen des „RELISH“-Projekts importiert werden (Broeder, Kemps-Snijders *et al.* 2010, S. 44). Weitere Ergänzungen erfolgen im Rahmen von „CLARIN-NL“. Hier ist auch die Component Metadata Infrastructure (CMDI) zu nennen, ein Framework zur Erstellung und zum Austausch von Metadaten, das auf ISOcat zurückgreift (Broeder, Kemps-Snijders *et al.* 2010; Broeder, Schonefeld *et al.* 2011). Aber auch die Werte und Kategorien des Stuttgart Tübingen Tag Set (STTS, Schiller *et al.* 1999) wurden in ISOcat importiert. Francopoulo, Declerck, Sornlertlamvanich *et al.* (2008) beschreiben die Arbeiten zum Aufbau eines DCS für morphosyntaktische Annotation. Die hierbei identifizierten Datenkategorien sind ebenfalls Bestandteil der aktuellen ISOcat-Implementierung.

Neben dem Web-Interface bietet ISOcat auch den Zugriff auf die DCR per RESTful Web Service an, um aus anderen Anwendungen heraus Datenkategorien einbinden zu können.[12]

8.3 Beurteilung

Die Möglichkeit einer standardisierten und über das Web zugänglichen Registrierung von Datenkategorien (sowohl für deren Nutzung als auch für die Erweiterung) ist prinzipiell zu begrüßen. Durch die Tatsache, dass ISOcat nicht nur die offizielle Referenzimplementierung der Norm ISO 12620:2009 darstellt, sondern zusätzlich im Rahmen des „CLARIN“-Projektverbunds entwickelt und eingesetzt wird (CLARIN 2009a), kann von einer großen Unterstützung in der wissenschaftlichen Community ausgegangen werden. Dazu trägt auch die Möglichkeit des Imports bereits bestehender Konzepte und Ontologien bei. Andere Standards können an diese zentrale Datenkategorisierungsregistrierung anknüpfen und sehen eine solche Verbindung als zentralen Baustein, darunter das Linguistic Annotation Framework (vgl. Kapitel 9) oder auch das Morpho-syntactic Annotation Framework und das Syntactic Annotation Framework (Kapitel 10 und 11). Damit ermöglicht ISO 12620:2009 die in Abschnitt 3.5 angesprochene Trennung von konzeptueller Ebene (Level) und Serialisierung (Layer), indem das Konzept vollständig losgelöst von der Annotation behandelt wird. Allen

[12] REST steht für REpresentational State Transfer und bezeichnet ein Architekturmodell, das in T. R. Fielding (2000) beschrieben wird.

genannten Spezifikationen kann durch die obligatorische Referenz auf die DCR eine Trennung von Level und Layer generell bescheinigt werden. Somit muss die Norm als ein Eckpfeiler für aktuelle und künftige Normungsarbeiten angesehen werden.

Allerdings kann auch hier davon ausgegangen werden, dass einzelne Projekte oder Gruppen mit den vorhandenen Datenkategorien (sprich: Konzepten) bzw. deren Definition in ISOCAT nicht einverstanden sind. Solche Konflikte lassen sich entweder durch Einrichtung von eigenen Untergruppierungen oder Ergänzung vorhandener Datenkategorien in der DCR lösen (sofern sie nicht konträr zu den bereits vorhandenen sind).

Aus texttechnologischer Sicht ist erfreulich, dass das RELAX-NG-Schema zur Definition des Austauschformats DCIF nicht nur Gebrauch von einigen nur in RELAX NG vorhandenen Konstrukten macht (z. B. einer Auswahl zwischen Attributen und Kindelementen, die nur hier möglich ist, da Attribute in RELAX NG Teil des Inhaltsmodells sind), sondern darüber hinaus auch eingebettete SCHEMATRON-Asserts (vgl. Abschnitt 3.4.2) nutzt. Damit ist die gewählte Constraint Language hinsichtlich der formalen Ausdrucksstärke und der technischen Merkmale optimal gewählt.

9 Das Linguistic Annotation Framework

Nach der Veröffentlichung der ersten XML-Version des Corpus Encoding Standards (XCES, vgl. Kapitel 7) wurde die Spezifikation zunächst nicht weiter entwickelt. Das Interesse an XCES war anfangs gering, auch wenn mit dem „American National Corpus“ (ANC) und später auch durch das *Institut für Deutsche Sprache* (*IDS*) durchaus umfangreiche Korpora im neuen Format annotiert bzw. nach XCES konvertiert wurden (vgl. Abschnitt 7.8). Dennoch konnte auch XCES nicht allen Wünschen an ein linguistisches Annotationsformat gerecht werden. Daher wurde im Rahmen von ISO/TC 37/SC 4 in der Arbeitsgruppe 1 (WG 1, „Basic descriptors and mechanisms for language resources“) die Spezifikation Linguistic Annotation Framework (ISO/FDIS 24612, LAF) erarbeitet.[1]

9.1 Hintergrund und formales Modell

Das Linguistic Annotation Framework stellt ein generalisiertes Konstrukt zur Darstellung von (multiplen) Annotationen dar, dessen Entwicklung verschiedene Ansätze und Annotationsformate der vergangenen Jahre berücksichtigt, darunter direkt die Arbeiten des Corpus Encoding Standard (vgl. Abschnitt 7) und des Annotation Graph (vgl. Abschnitt 3.3.3.2). LAF wird bereits seit August 2002 entwickelt (im September 2007 wurde ein Antrag auf Verlängerung der ursprünglich auf vier Jahre angelegten Entwicklungszeit gestellt), die öffentlich einsehbaren Versionen Ide (2006) und ISO/CD 24612 verdeutlichen die unterschiedlichen Stadien der Entwicklung. Aktuell ist die Spezifikation im Status eines Final Draft International Standard (ISO/FDIS 24612, vgl. auch Abschnitt 2.1.1 und die Ausführungen in Abschnitt 9.4). In diesem Zeitraum hat es einige Änderungen gegeben, die teilweise der Art und Weise geschuldet sind, wie Internationale Standards entwickelt werden, aber auch durch Anpassungen an neue

[1] Wie aus der Literaturreferenz ersichtlich, ist die Normierung des Linguistic Annotation Framework zum Zeitpunkt dieser Arbeit noch nicht abgeschlossen. Da die Veröffentlichung des FDIS kurz vor Fertigstellung dieser Arbeit erfolgte, beziehen sich weite Teile der Ausführungen dieses Kapitels noch auf die Vorversion ISO/DIS 24612. Aktuelle Entwicklungen werden in Abschnitt 9.4 thematisiert.

Entwicklungen im Bereich der Strukturierung von linguistischen Informationen. Das Framework erhebt folgenden Anspruch:

> This International Standard specifies a framework for representing linguistic annotations of language data such as corpora, speech signal, video, etc. This framework includes an abstract data model and an XML serialization of that model for representing annotations of primary data. The serialization serves as a pivot format to allow annotations expressed in one representation format to be mapped onto another. (ISO/FDIS 24612, S. 5)

In diesem Sinne soll LAF als *Common Ground*, als gemeinsamer Nenner, angesehen werden, der grundlegende Prinzipien bei der Gestaltung und Ausarbeitung linguistischer Beschreibungsschema vorgibt. Wichtig ist hervorzuheben, dass LAF im Gegensatz zu Spezifikationen wie den TEI GUIDELINES kein Annotationsschema (also ein Tagset) vorgibt; die zukünftige Norm definiert ein abstraktes Datenmodell (*Data Model*), die Segmentierung (*Base Segmentation*), mögliche Serialisierungsformate (zu unterscheiden zwischen der Serialisierung in der vom Anwender definierten Dokumentenformat, *Document Form*, und dem Austauschformat (*Pivot Format* oder auch *Dump Format*) als Teil des Standards, vgl. Abschnitt 9.2), sowie ein API zur Manipulation des Datenmodells. Folgende Anforderungen sind in ISO/DIS 24612, S. 4f. spezifiziert:[2]

- Adäquatheit der Ausdrucksstärke
  LAF soll alle nötigen (und nur diese) Mittel zur Repräsentation aller linguistischer (und möglicherweise anderer) Daten zur Verfügung stellen.

- Unabhängigkeit von Medientypen
  Alle multi-medialen Formate sollen – unter Zuhilfenahme existierender (oder neuer) Standards – unterstützt werden.

- Formale Eindeutigkeit
  Die durch LAF formalisierte Repräsentation muss über eine formale Semantik verfügen (inkl. eines Inventars logischer Operationen, die wiederum die abstrakte Semantik des Datenmodells definieren).

- Inkrementalität
  LAF soll verschiedene Stadien von linguistischer Analyse (sowohl auf der Eingabe- als auch Ausgabenseite) unterstützen, d. h., auch unterspezifizierte bzw. ambige Daten sollen repräsentiert und – in einem weiteren Schritt – analysiert, erweitert und verarbeitet werden können.[3]

- Einheitlichkeit der Beschreibung
  Die Repräsentation mittels LAF soll – unabhängig von den betrachteten Daten und deren Skalierbarkeit – mit den gleichen grundlegenden basalen Einheiten möglich sein.

[2] Diese Anforderungen sind nicht mehr explizit in ISO/FDIS 24612 aufgeführt, es kann aber davon ausgegangen werden, dass sie implizit noch Bestand haben.

[3] Dieser Punkt lässt sich als direkte Referenz zu der mit dem CORPUS ENCODING STANDARD eingeführten Verarbeitungsweise der Up-translation bzw. Up-Conversion verstehen, vgl. Abschnitt 7.2.

- Offenheit
  LAF soll eine Repräsentation unabhängig von bestimmten linguistischen Theorien ermöglichen.

- Erweiterbarkeit
  Schnittstellen für die Entwicklung und Implementation von Erweiterungen für die zentrale Datenkategorisierung müssen zur Verfügung stehen.

- Lesbarkeit
  XML-Auszeichnungssprachen sind nicht zwangsläufig menschenlesbar. Auf LAF beruhende Repräsentationen sollen es sein – zumindest in Bezug auf Erstellung und Überarbeitung.

- Explizitheit in Bezug auf die Verarbeitung
  Annotationsschemata sollen explizit sein in Bezug auf die darin enthaltenen Relationen – und damit unabhängig von spezifischen Software-Lösungen.

- Widerspruchsfreiheit
  Verschiedene Beschreibungsmechanismen sollen nicht für den gleichen Typ von Informationen genutzt werden können – eine Eindeutigkeit der Beschreibung soll so sichergestellt werden.

Daraus ergeben sich (un-)mittelbar u. a. die folgenden Design-Prinzipien (vgl. ISO/-DIS 24612, S. 5):

1. Datenmodell und Serialisierung sind eindeutig trennbar, lassen sich aber in beiden Richtungen aufeinander abbilden.

2. Das Datenmodell ist möglichst sparsam, allgemein und formal eindeutig und unterscheidet zwischen Form und Inhalt.

3. Das Dokumentenformat ist weitgehend vom Anwender festlegbar, die Abbildung zwischen Serialisierung und Datenmodell erfolgt über das Pivot Format, wobei für die Abbildung entweder ein XML-Schema bzw. ein funktional äquivalenter Grammatikformalismus oder ein Stylesheet herangezogen werden kann.[4] Ein Annotationsschema (im Sinne der Document Form) muss isomorph zum LAF-Datenmodell sein – entweder direkt oder über eine Abbildung auf das Dump Format (vgl. Abschnitt 9.2).

4. Einzelne Annotationsebenen sind sowohl von anderen als auch den Primärdaten trennbar (im Sinne einer Standoff-Annotation).

Das Datenmodell besteht aus einem Referenzierungsmechanismus (*Referential Structure*) in Form eines gerichteten Graphen (vgl. Abschnitt 3.3.3), um Primärdaten und Standoff-Annotation miteinander zu verknüpfen und einer Merkmalsstrukturbeschreibung (vgl. Kapitel 6) zur Speicherung der Annotation als solche (das aktualisierte

[4] Der Standard lässt hier Raum für Interpretationen, in wieweit eine der anderen in Abschnitt 3.4 diskutierten Constraint Languages als „funktional äquivalent“ angesehen werden kann.

Datenmodell aus ISO/FDIS 24612 ist in Abbildung 9.3 auf Seite 211 dargestellt). Formal beschreiben Ide und Suderman (2007, S. 2f.) das Datenmodell wie folgt:

> A data model for annotations based on directed graphs defined as follows: A graph of annotations G is a set of vertices $V(G)$ and a set of edges $E(G)$. Vertices and edges may be labeled with one or more features. A feature consists of a quadruple (G', VE, K, V) where, G' is a graph, VE is a vertex or edge in G', K is the name of the feature and V is the feature value.

Mittels LAF annotierte linguistische Daten bestehen aus den Primärdaten, der bereits erwähnten Segmentierung, einer beliebigen Anzahl an Annotationen, die über die Segmentierung auf die Primärdaten abgebildet werden können, und Metadaten (*Header Document*). Die Primärdaten sind nicht zwingend die Originaldokumente (z. B. im Falle nicht digital vorliegender Rohdaten), sondern zum Zwecke der Annotation erstellte Dateien, auf die nur lesend zugegriffen wird (um die Referenz mittels Segmentierung eindeutig zu ermöglichen). Die Segmentierung erfolgt über Zeichenpositionen (bei textuellen Daten standardmäßig ein Zeichen pro Position bei Annahme von UTF-8-Kodierung), beginnend bei 0 (links vom ersten Zeichen, vgl. auch Tabelle 13.1 auf Seite 249 und Abbildung 13.1 auf Seite 244).[5] Diese Positionen werden als Knoten eines virtuellen Graphen aufgefasst, der durch Kanten zwischen den einzelnen Buchstaben (den Knoten) verbunden ist. Sie stellen die Blätter des Graphen dar. Segmente sollen nicht überlappend angelegt werden, als Beispiel wird im Standard ein Segment von 5 bis 9 und ein zweites von 7 bis 15 angegeben. Diese sollten als drei Segmente (5–7, 7–9, 9–15) dargestellt werden, Graphenrelationen zwischen den beiden ersten und dem zweiten und dritten Knoten stellen die ursprüngliche Abdeckung über die Primärdaten her. Diskontinuierliche Segmente werden ebenfalls über entsprechende Referenzierungen (via drei Knoten – je einer pro Segment und ein zusätzlicher *Dummy Node* zur Verknüpfung der beiden) realisiert. Mehrere Segmentierungen sind ebenso möglich wie mehrere Annotationsebenen, die auf eine Base Segmentation verweisen, es besteht also eine *n*:*m*-Relation. Formal definieren Ide und Suderman (2007, S. 2f.) die Segmentierung wie folgt:

> A *base segmentation* of primary data that defines edges between virtual nodes located between each „character“ in the primary data. [...] The resulting graph G is treated as an *edge graph* G' whose nodes are the edges of G, and which serve as the leaf („sink“) nodes. These nodes provide the base for an annotation or several layers of annotation. Multiple segmentations can be defined over the primary data, and multiple annotations may refer to the same segmentation.

Sowohl für die Primärdaten als auch die Annotationsebenen sind eigene Header vorgesehen. In ISO/CD 24612 ist noch die Rede von einem *Corpus Header*, der in der

[5] Zu beachten ist hierbei, dass in der UTF-8-Kodierung ein Zeichen nicht gleichbedeutend mit einer festen Anzahl an Bytes ist: „In UTF-8, a character may be expressed with one, two, three, or four bytes, and the relationship between those byte values and the code point value is more complex.“ (Unicode 6.0.0, S. 26).

aktuellen Fassung nicht mehr erwähnt wird. Das *Header Document* der Primärdatendatei folgt den Metadaten des XCES-Standards (vgl. Abschnitt 7), im Gegensatz zu diesem sind aber alle Metadaten zwingend in einer eigenen Datei gespeichert. Primär- und Metadatendatei(en) sind nicht miteinander verknüpft, da erstere üblicherweise nicht als XML-Instanz vorliegen, allerdings dient der *Primary Data Document Header* u. a. dazu, den eigentlichen Primärdaten einen eindeutigen Bezeichner *PID* zuzuweisen.

Das Header Document der jeweiligen Annotationsdateien (*Annotation Document Header*) enthält folgende Elemente:

1. Ein `dependencies`-Element zur Gruppierung von Abhängigkeiten zwischen verschiedenen Dateien. Zu diesem Zweck verfügt jedes der nachfolgend genannten obligatorischen Kindelemente über ein `loc`-Attribut zur Referenzierung der entsprechenden Datei (mittels *PID*). Die unterhalb von `dependencies` auftretenden Elemente sind
   - `primaryData` zur Referenz auf die Primärdaten.
   - `baseSegmentation` verweist auf die Segmentierung (direkt über die Angabe einer entsprechenden Datei oder indirekt über die Angabe einer Annotationsebene, die diese nutzt).
   - `baseAnnotation` zur Referenz auf die Annotationsdatei, auf die sich die Annotation, zu der das jeweilige Header Document gehört, bezieht. Unter Umständen ist das die Segmentierung – in diesem Fall sind die Werte des `loc`-Attributs identisch zu dem des `baseSegmentation`-Elements.
   - `dependsOn` (optional) zur Referenzierung anderer Annotationsebenen, von denen die aktuelle abhängig ist, während `referencedBy` zum Verweis auf eine andere Annotation genutzt wird, die Knoten der aktuellen referenziert.
2. Darüber hinaus muss zumindest eines der beiden folgenden Elemente Teil der Metadaten sein:
   - `externalDocumentation` zur Referenz auf eine Dokumentation (ebenfalls via `loc`-Attribut), die verschiedene Prozesse der Korpuserstellung beschreibt.
   - Mittels `typeSystemDescription` kann eine formale Spezifikation der Annotation (Inhalt und Struktur) entweder über ein `loc`-Attribut oder über Kindelemente erfolgen, letzte sind im Listing 9.1 aufgeführt und orientieren sich an der Typsystem-Beschreibung der Unstructured Information Management Architecture (UIMA, vgl. Apache UIMA Development Community 2010, S. 5ff.). Ein Typsystem definiert die Typen von Objekten, die in einer Annotation enthalten sind (in XML also die Elemente und Attribute) und deren Merkmale.

Anmerkung: Das in Listing 9.1 gezeigte Header Document basiert auf ISO/DIS 24612. Die hier zu findenden Informationen sind sehr allgemein gehalten (beispielsweise in Bezug auf Element- und Attributnamen) und verweisen auf XCES (vgl. Kapitel 7), so dass die Serialisierung nicht zwangsläufig korrekt sein muss.

Listing 9.1: Fiktiver Header für ein LAF Annotation Document

```
<?xml version="1.0" encoding="UTF-8"?>
<header>
 <!-- [...] -->
 <dependencies>
  <primaryData loc="p1"/>
  <baseSegmentation loc="s1"/>
  <baseAnnotation loc="b1"/>
 </dependencies>
 <externalDocumentation loc="d1"/>
 <typeSystemDescription>
  <name>SynF</name>
  <description>Syntaktische Merkmale</description>
  <typeDescription>
   <name>net.xstandoff.types.syn.sub</name>
   <description>Syntaktische Merkmale von Substantiven</description>
   <supertypeName>net.xstandoff.annotations</supertypeName>
   <features>
    <featureDescription>
     <name>Genus</name>
     <values>Maskulinum, Femininum, Neutrum</values>
     <description>Grammatisches Geschlecht</description>
    </featureDescription>
    <!-- [...] -->
   </features>
  </typeDescription>
 </typeSystemDescription>
</header>
```

Bei der Verwendung von Typen- und Merkmalsdefinitionen wird empfohlen, auf vorhandene Standards zurückzugreifen, hier wäre vorrangig die DATA CATEGORY REGISTRY (DCR, Kapitel 8) zu nennen. Eine Typen- oder Merkmalsdefinition besteht sowohl aus dem Attribut bzw. dem Merkmal als solchem (z. B. syntaktische Kategorie) und den möglichen Werten, die dieses annehmen kann (Nomen, Verb, etc.). Dabei wird unterschieden zwischen der konzeptuellen Ebene und der konkreten Instanziierung (z. B. in Form eines XML-Elements, -Attributs oder einer anderen Notation) – ähnlich wie bei der Differenzierung zwischen Level und Layer in Abschnitt 3.5. Dabei können die allgemeinen Konzepte entweder aus einer bestehenden DCR (wie dem in Abschnitt 8.2 genannten ISOCAT) stammen oder durch den Anwender definiert werden (DATA CATEGORY SPECIFICATION, DCS). In diesem Fall stellt die DCR die Abbildung zwischen den in ihr gespeicherten Konzepten und der konkreten Instanziierung des Anwenders her (vgl. auch Ide und Romary 2004b, S. 218; sowie Ide und Romary 2004a).

Im Laufe der mehrjährigen Entwicklung von LAF hat sich vor allem die Strukturierung der Metadaten (also die Hierarchie und Bezeichnung innerhalb der entsprechenden Header) mehrfach geändert. Das wird z. B. deutlich, wenn man den Committee Draft (CD, ISO/CD 24612) mit dem Draft International Standard (DIS, ISO/DIS 24612) vergleicht. Auch sind Unterschiede zum XCES- und TEI-Header feststellbar: So sind alle im Listing 9.1 aufgeführten Elemente *nicht* Teil der entsprechenden anderen Spezifikationen, im Listing nicht gezeigte Metadaten wie z. B. das Element `fileDesc`, das u. a. Angaben zum Titel, der Version, Dateigröße und Quelle enthält, sollten daher in Kombination mit den im Standard aufgeführten genutzt werden, wobei es auch zu Dopplungen

mit Metaangaben zur Primärdatendatei kommen kann, die durchaus beabsichtigt sind (vgl. ISO/DIS 24612, S. 7f.). Auch in der aktuellsten Version, FDIS, ISO/FDIS 24612, ist der Header erneut geändert worden: Der Header für die Primärdaten (Primary Data Document Header) nutzt weiterhin Elemente aus XCES und ist tabellarisch in ISO/FDIS 24612, S. 11ff. aufgeführt. Der Annotation Document Header ist hier als Teil der Annotationsdatei vorgesehen und wird unterhalb des `graphHeader`-Elements gespeichert. Er besteht aus den Elementen `labelsDecl`, `dependencies`, `annotationSpaces` (vgl. Listing 9.4).

9.2 GrAF: ein Graphen-basiertes Format zur Annotation linguistischer Daten

Graphen-basierte Formate erlauben im Gegensatz zum Großteil der bisher diskutierten Annotationsformate die Darstellung mehrerer Annotationsebenen in Form eines einzigen Graphen, d. h., nicht nur einzelne Ebenen können in Form eines Graphen gespeichert werden, sondern auch die Kombination bzw. Unifikation verschiedener Ebenen, was die Verarbeitung und Analyse stark vereinfacht (vgl. Ide und Suderman 2007, und Abschnitt 3.3.3). Ein weiterer Vorteil ist die Möglichkeit, mittels einer Vielzahl bestehender Werkzeuge zur Verarbeitung von Graphen, Relationen zwischen einzelnen Annotationsebenen zu ermitteln und Aussagen über die Unifikation zu machen.

9.2.1 Aufbau

Mit dem Graph Annotation Format (GrAF) ist eine XML-Serialisierung des Pivot (oder Dump) Formats, das als Mittler zwischen Datenmodell und vom Anwender festgelegten Dokumentenformat dient, Teil von ISO/DIS 24612. Darüber hinaus kann GrAF als Zwischenformat zwischen den verschiedenen proprietären Annotationsformaten dienen (ISO/DIS 24612, S. 9), oder wie es Ide und Suderman (2010, S. 32) ausdrücken: „GrAF is intended to serve as the lingua franca for data and annotations used in processing systems such as GATE and UIMA."

Damit reduziert dessen Einsatz die Anzahl notwendiger Abbildungen (z. B. in Form von XSLT-Transformationen) auf $2n$, an Stelle von $n^2 - n$ (bei direkter wechselseitiger Konvertierung aller n Formate, vgl. auch die Abbildung 1 in ISO/DIS 24612, S. 9).

Sinnvollerweise kommt der Anwender nicht direkt mit dem Dump Format in Berührung, sondern formalisiert lediglich die Abbildung zwischen den verschiedenen Formaten (Ide und Romary 2007, S. 272f.). Neben den beiden genannten Abbildungsleistungen unterstützt GrAF zusätzlich die Vereinigung von Annotationen und die Analyse linguistischer Phänomene (mittels mehrerer verschiedener Auszeichnungen, z. B. als Ausgabe verschiedener Ressourcen).

Eine GrAF-Dokumentgrammatik (erstellt mittels der Constraint Language XML Schema, vgl. Abschnitt 3.4) ist Teil des (informativen) Anhangs in ISO/DIS 24612. Das Wurzelelement einer GrAF-Instanz ist das Element `graph` aus dem Namensraum `http://www.tc37sc4.org/graf/v1.0.6b` (vgl. Abbildung 9.1).[6]

[6] Darüber hinaus existiert unter der Adresse `http://www.xces.org/ns/GrAF/0.99/` ein „RDDL (Re-

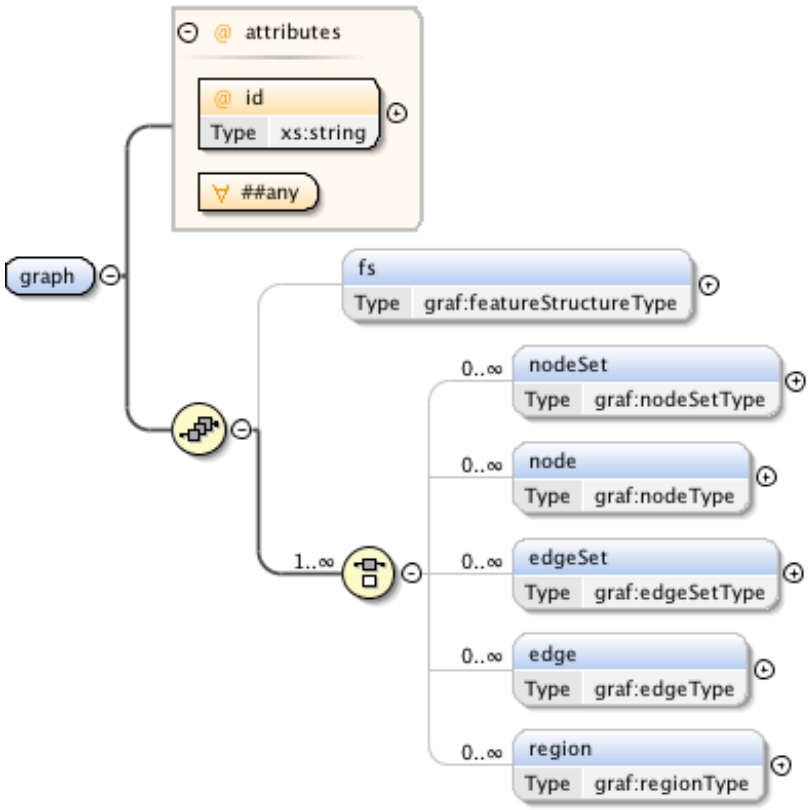


Abbildung 9.1: Das Wurzelelement graph

Die wesentlichen Kindelemente sind node und edge, die entsprechend Knoten und Kanten des Graphen repräsentieren. Sowohl Knoten als auch Kanten lassen sich mittels der Elemente nodeSet bzw. edgeSet gruppieren und können mit Annotationen assoziiert werden – wobei Kanten keine expliziten Annotationen, sondern eher Informationen über diese enthalten, so dass Auszeichnungen den Knoten zugeordnet werden.[7] Es ist nicht ganz ersichtlich, ob diese Regelung eine „Muss-“ oder „Kann“-Bestimmung darstellt, allerdings erlaubt das (nicht-normative) XSD, das Teil von ISO/DIS 24612 ist, *kein* Unterelement as (oder andere Unterelemente) als Kind von edge.

Eine Annotation ist eine Merkmalsstruktur, die durch einen Graph repräsentiert

source Directory Description Language) Directory for the Graph Annotation Framework 0.99“. Hier lassen sich ebenfalls Schemata für GRAF herunterladen, ein als „normative“ bezeichnetes RELAX-NG-Schema sowie ein als „non-normative“ bezeichnetes XSD. Beide weichen sowohl vom verwendeten Zielnamensraum (*targetNamespace*) als auch von den vorhandenen Elementen und den erlaubten Strukturen von dem hier verwendeten XSD ab. Es ist unklar, ob die Angabe „0.99“ als Versionsnummer zu verstehen ist, es ist allerdings – aufgrund des Dateidatums aus dem Jahr 2009 – davon auszugehen, dass diese Version älter als die in ISO/DIS 24612 enthaltene ist, deren Namensraum die Angabe „v1.0.6b“ enthält.

[7] Das gilt auch für Annotationen, die üblicherweise gelabelte Kanten haben, wie z. B. Dependenzanalysen (Ide und Suderman 2007, S. 3).

wird und nach ISO 24610-1:2006 enkodiert ist.[8] Sie wird einem Knoten im Graphen zugeordnet, indem sie als dessen Kindelement in der GRAF-XML-Instanz realisiert ist (unterhalb des `as`-Elements, vgl. Abbildung 9.2).

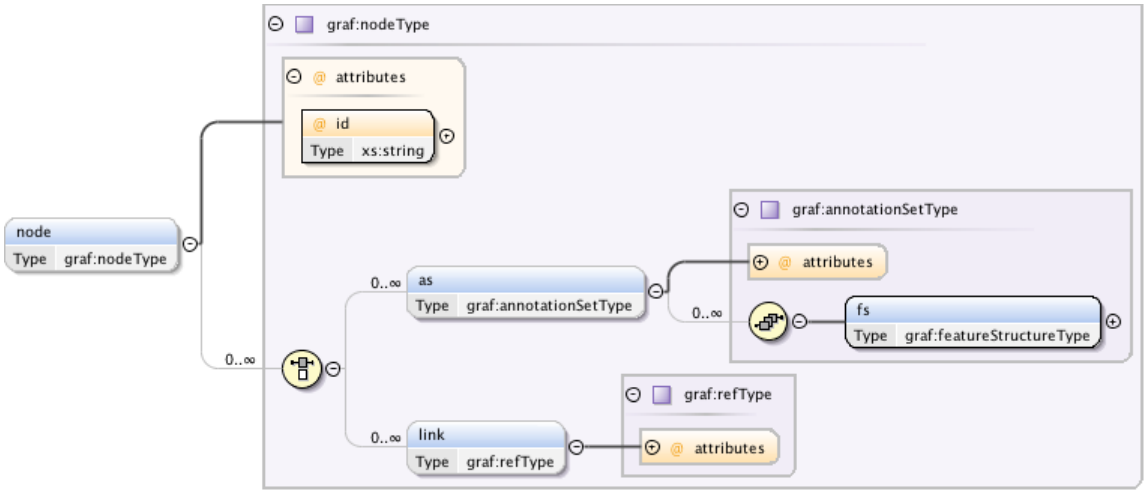


Abbildung 9.2: Das Element node

Das Element `as` strukturiert ein sogenanntes *Annotation Set*, das über ein optionales `type`-Attribut näher spezifiziert werden kann und wiederum selbst aus einer Merkmalsstruktur (*Feature Structure*, `fs`) besteht. Eine Merkmalsstruktur beinhaltet ein einzelnes Merkmal (`f`) oder rekursiv weitere `fs`-Elemente und wird über ein optionales type-Attribut weiter spezifiziert. Ein *Feature* hat einen Namen (in Form des `n`-Attributs) und einen Wert (`v`). Beide Attribute sind vom Datentyp `xs:string`, die Spezifikation weist darauf hin, dass für binäre Merkmalswerte diese über eine PID (oder alternative Verknüpfungsmechanismen) referenziert werden sollen – GRAF-Instanzen sollen nur textuell kodierte Informationen beinhalten.

In ISO/CD 24612 sind Metadaten in Form eines `header`-Elements noch Bestandteil einer GRAF-Instanz. Da in ISO/DIS 24612 je ein separates Header Document pro Annotationseinheit vorgesehen ist, kann davon ausgegangen werden, dass das auch für die GRAF-Serialisierung gilt. Ebenfalls im Laufe der Entwicklung entfernt wurde das Element `sink`, das in Ide und Suderman (2007, S. 3) noch zur Strukturierung einer Basissegmentierung (in Form von Knoten ohne ausgehende Kanten) erwähnt wird.

Die Listings 9.2 und 9.3 zeigen die logische Dokumentstruktur und die Annotation der Verbal-Chunks desselben Primärdatums (in gekürzter Darstellung). Es handelt sich um einen Artikel aus dem SLATE-Magazin, mit dem Titel „Harmonic Convergences“ (Autoren: Michael Hirschorn und Ellen Willis), der Teil des „Open American National Corpus“ (vgl. Kapitel 7 und Abschnitt 9.3) ist. Da die Ursprungsdateien nicht dem aktuellen Stand der Spezifikation entsprechen (lt. Metadaten wurden die letzten Änderungen im Jahr 2007 durchgeführt), wurden die Beispielinstanzen an die Ausführungen aus ISO/DIS 24612 angepasst.[9]

[8] Zusätzlich sieht die Spezifikation eine eigene, von ISO 24610-1:2006 abweichende Serialisierung vor.
[9] Die Originaldateien sind nach Download und Extraktion des OANC-Archivs zu finden im Order `data`

Listing 9.2: GrAF-Instanz der logischen Dokumentstruktur

```
<?xml version="1.0" encoding="UTF-8"?>
<graph id="g_log">
 <region id="n-r0" a="0 3662"/>
 <node id="n-n0">
  <link to="n-r0"/>
  <as type="xces">
   <fs type="cesDoc">
    <f n="xmlns:xlink" v="http://www.w3.org/1999/xlink"/>
    <f n="xmlns:xsi" v="http://www.w3.org/2001/XMLSchema-instance"/>
    <f n="xmlns" v="http://www.xces.org/schema/2003"/>
    <f n="version" v="1.0.4"/>
   </fs>
  </as>
 </node>
 <region id="n-r3" a="38 59"/>
 <node id="n-n3">
  <link to="n-r3"/>
  <as type="xces">
   <fs type="head">
    <f n="type" v="headline"/>
    <f n="level" v="1"/>
   </fs>
  </as>
 </node>
 <region id="n-r4" a="86 1842"/>
 <node id="n-n4">
  <link to="n-r4"/>
  <as type="xces">
   <fs type="p">
    <f n="id" v="p1"/>
   </fs>
  </as>
 </node>
 <region id="n-r5" a="100 105"/>
 <node id="n-n5">
  <link to="n-r5"/>
  <as type="xces">
   <fs type="hi">
    <f n="rend" v="ital"/>
    <f n="type" v="title"/>
   </fs>
  </as>
 </node>
 <!-- [...] -->
</graph>
```

Die in Listing 9.2 gezeigte Annotation der logischen Dokumentstruktur kommt ohne `edge`-Elemente aus, besitzt also keine Kanten im Graph. Zu sehen sind jeweils die Definition einzelner Bereiche (Element `region`), die mittels `a`-Attributs ein oder mehrere Ganzzahlwerte zur Speicherung von Offset-Positionen im Primärdatum die Basissegmentierung vornehmen, auf die die Annotationen per `link`-Element (genauer: über dessen `to`-Attribut) verweisen. Gut sichtbar sind die Merkmalsstrukturen, die als Kind des `as`-Elements die Informationen der klassischen Inline-Annotation standardisiert

`/written_1/journal/slate/1` und tragen den Namen Article247_4-logical.xml bzw. Article247_4-vp.xml.

speichern. So werden in Zeile 38ff. Angaben über eine Hervorhebung (`type="hi"`) gemacht, die zum einen in der Formatierung durch Wahl eines Kursivschnitts (`n="rend" v="ital"`) und zum anderen semantisch ausgedrückt werden (`n="type" v="title"`). Da sich Merkmalsstrukturen beliebig tief rekursiv verschachteln lassen, lassen sich damit elegant beliebig viele Angaben in arbiträrer Hierarchie festhalten. Verloren geht allerdings die Angabe, in wieweit die Merkmale in der ursprünglichen Form als Element oder Attribut gespeichert wurden – die Möglichkeit einer eindeutigen Rekonstruktion ist damit nicht mehr gegeben. Die in Listing 9.3 dargestellte Annotation von Verbal-Chunks nutzt keine eigene Segmentierung (in Form von `region`-Elementen), verwendet dafür aber Kanten im Graph und wurde daher als zusätzliches Beispiel für den Aufbau von GRAF-Instanzen herangezogen.

Listing 9.3: GrAF-Instanz der Verbal-Chunks

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graph id="g_vp">
3   <node id="vp-n0">
4    <as type="xces">
5     <fs type="VG">
6      <f n="voice" v="active"/>
7      <f n="tense" v="SimPre"/>
8      <f n="type" v="FVG"/>
9     </fs>
10   </as>
11  </node>
12  <!-- [...] -->
13  <node id="vp-n53">
14   <as type="xces">
15    <fs type="VG">
16     <f n="voice" v="active"/>
17     <f n="tense" v="Inf"/>
18     <f n="type" v="NFVG"/>
19    </fs>
20   </as>
21  </node>
22  <!-- [...] -->
23  <edge id="lnk69" from="vp-n53" to="penn-n426"/>
24  <!-- [...] -->
25 </graph>
```

Die Kante (Zeile 23) verbindet zwei Knoten, im Beispiel den Knoten mit dem Bezeichner *vp-n53* aus der lokalen Datei mit einem Knoten *penn-n426*, der in einer anderen Datei definiert ist. Laut Spezifikation handelt es sich beim Wert der Attribute `from` und `to` um „ID values of the source and destination vertices“ (ISO/DIS 24612, S. 13). Im XSD, durch das GRAF im informativen Teil des Standards definiert wird, sind beide Attribute mit dem Datentyp `xs:string` (Zeichenkette) deklariert. Somit macht GRAF keinen Gebrauch von den Integritätsmerkmalen, die dieser Grammatikformalismus zur Verfügung stellt (vgl. Abschnitt 3.4). Dies ist allerdings nachvollziehbar, da sowohl der in XML-DTD vorhandene Token-Typ `ID` als auch das exklusiv in XML SCHEMA zu findende `xs:key`-Element nur eine dokumentweite Überprüfung der Integrität ermöglichen – im Beispiel würde ein validierender Parser die in Zeile 23 aufgeführte Kante monieren, da ein die Zeichenkette *penn-n426* nicht als Wert eines Attributs vom Typ `xs:ID` im

Dokument zu finden ist. Es irritiert etwas, dass das (zugegebenermaßen nur im informativen Anhang von ISO/DIS 24612, aufgeführte) XSD darüber hinaus `id`-Attribute je nach übergeordnetem Element unterschiedlich deklariert: Bei den Elementen `graph`, `nodeSet`, `node` und `as` sind Attribute diesen Namens vom Datentyp `xs:string`, die den Elementen `edge` und `edgeSet` zugeordneten vom Typ `xs:ID`. Weiterhin ist unklar, warum nicht das global zur Verfügung stehende `xml:id`-Attribut eingebunden wurde, anstatt ein eigenes Attribut lokal zu deklarieren.

Zwar erleichtert die strikte Trennung in separate Dateien für jede Annotationsebene (mit zusätzlichem Header Document) das Auffinden und Bearbeiten einer spezifischen Auszeichnung, allerdings gestaltet sich der Austausch des gesamten Korpuseintrags gegebenenfalls aufwändiger (mindestens eine Primärdatendatei – je nach Medientyp – nebst Metadatendatei, x Dateien für Annotationsebenen mit y Header Documenten, wobei x=y).

9.2.2 Zusammenfassen von Annotationen

Ein Vorteil von GRAF-Instanzen ist, dass sie sich mit Hilfe von allgemeinen Graphenalgorithmen zusammenfassen lassen. Der dazu in ISO/DIS 24612, S. 17 aufgeführte Algorithmus (in Pseudocode-Notation) ist in Abbildung 1 dargestellt.[10] Ausgangspunkt ist der Graph G, der die Basissegmentierung beschreibt.

Given: a graph G:
  for all graph of annotations G_p **do**
    for all vertex v_p in G_p **do**
      if v_p is not a leaf in G_p **then**
        add v_p to G
      end if
    end for
    for all edge (v_i, v_j) in G_p **do**
      if v_j is a leaf in G_p **then**
        find corresponding vertex $v_g \in G$
        add a new edge (v_i, v_g) to G
      else
        add edge $(v\,i, v\,j)$ to G
      end if
    end for
  end for

Algorithmus 1: Kombination mehrerer GRAF-Annotationsebenen

Dieser Algorithmus kann sowohl auf Annotationen angewandt werden, die sich auf die gleichen Primärdaten beziehen, als auch auf Annotationsebenen, die andere referenzieren. Der resultierende Graph kann anschließend über entsprechende Algorithmen

[10] Die Notation wurde aus Gründen der formalen Eindeutigkeit geringfügig angepasst.

weiter minimiert werden, z. B. indem Mehrfachvorkommen identischer Knoten zusammengefasst und weitere überflüssige Knoten (*Dummy Nodes*) entfernt werden – die Spezifikation bezeichnet den Gesamtvorgang als „Merge“. Wird auf die Minimierung verzichtet, sprechen die Autoren von „Add“. Für beide Varianten der Verarbeitung stellt das GRAF-API Methoden zur Verfügung (vgl. nächster Abschnitt).

9.3 LAF und GrAF in der Anwendung

In ISO/DIS 24612 selbst werden – neben der im vorherigen Abschnitt aufgeführten Möglichkeit der Kombination mehrere GRAF-Instanzen – als weitere Verwendungsmöglichkeiten Visualisierungen (z. B. mittels vorhandener Software zur Graphendarstellung, die Spezifikation nennt hier GRAPHVIZ[11]) und Methoden der allgemeinen Graphenmanipulation genannt. Diese Darstellungen sind äußerst vage gehalten: In Bezug auf eine Konvertierung zwischen Dateien des genannten GRAPHVIZ und GRAF-Instanzen wird nur allgemein auf die iterierende Verarbeitung der Knoten und Kanten mit anschließender Ausgabe einer „geeigneten“ Repräsentation verwiesen: „Generating the input files to GraphViz involves simply iterating over the nodes and edges in the graph and printing out a suitable string representation.“ (ISO/DIS 24612, S. 17).[12]

Auch die Ausführungen zu Traversierung oder Graphenmanipulation (beispielsweise Färbung) zum Zwecke statischer Auswertungen sind nur rudimentär vorhanden, so dass dieser Abschnitt der Spezifikation sicherlich besser im informativen Teil aufgehoben wäre. Das lässt sich dadurch erklären, dass die Abschnitte „Merging annotations“ und „Using the graphs“ aus Ide und Suderman (2007) übernommen wurden.[13]

Obwohl sich der Standard aktuell noch in der Entwicklung befindet (wenn auch in einem weit vorangeschrittenem Stadium), sind Anwendungen, die sich auf LAF und GRAF stützen, bereits im Einsatz. Dabei ist generell zu unterscheiden zwischen Arbeiten, die nur das Datenmodell von LAF nutzen und solchen, die auch eine zu GRAF kompatible Serialisierung verwenden.

Die aktuelle Version des „Open American National Corpus“ (OANC, vgl. auch Kapitel 7) wurde mittels GRAF annotiert. Da eine frühere Fassung des „Second Release“ ursprünglich mit Hilfe von XCES ausgezeichnet wurde, sind die Angaben über das Annotationstagset teilweise widersprüchlich: Auf der Startseite wird betont, dass zur Auszeichnung GRAF eingesetzt wird:

> The full 15 million word OANC is now available in GrAF format. GrAF is the ISO standard serialization format for standoff annotations over linguistic data. [...] Please consult the latest draft of the [ISO/TC 37/SC 4] Linguistic Annotation Framework for details about GrAF.

[11] Nähere Informationen und Download unter der URL `http://www.graphviz.org`.

[12] Ein entsprechendes Format für den GRAPHVIZ-Import stellt die Beschreibungssprache DOT dar, die wie GRAPHVIZ auch von *AT&T* und den *Bell-Labs* entwickelt wurde. Eine Einführung findet sich in Gansner *et al.* (2006) oder auch unter `http://www.graphviz.org/doc/info/lang.html`, zuletzt abgerufen am 19.04.2012.

[13] In ISO/FDIS 24612 fehlen die Ausführungen entsprechend.

Dagegen verweist die Unterseite zum OANC darauf, dass auf die Spezifikation der zweiten Ausgabe des ANC zurückgegriffen wurde (beschrieben in Ide und Suderman 2006b; Ide und Suderman 2006a); diese wiederum beruhen ebenfalls auf XCES:

> The ANC Second Release uses the proposed XCES Markup for Standoff Annotations. Each logical document in the ANC is conceptually a single XML document that conforms to the XCES xcesDoc.xsd schema. Physically, the primary data and its annotations are stored in multiple XML documents that form a directed graph referencing regions of primary data (and potentially, regions defined over other annotations as well).

Diese Irritation basiert darauf, dass es vom „Second Release“ des OANC zwei Versionen gibt, wobei die GrAF-annotierte Fassung nicht die aktuelle Version der zukünftigen Norm verwendet: Die zum Zwecke der Dokumentation zum Download bereit gestellte Version der Spezifikation des Linguistic Annotation Frameworks ist veraltet (ISO/CD 24612).[14] Umso ärgerlicher ist folgende Aussage der Ankündigungsmail, die am 17. Februar 2011 durch Nancy Ide an verschiedene Verteiler ging:

> The 15 million word Open American National Corpus (OANC) is now available in the finalized version of the ISO Linguistic Annotation Framework (LAF) Graph Annotation Format (GrAF). Formerly, the OANC was available in an early version of the LAF specification for a pivot format for standoff linguistic annotations.

Es ist allerdings davon auszugehen, dass zukünftige Versionen des (O)ANC die finale Notation von GrAF nutzen werden, da die Erstellung des Korpus und die Entwicklung der Norm eng Hand in Hand zusammengehen:

> The ANC is the poster child for the specifications under development by the International Standards Organization (ISO) [TC 37/SC 4] (Language Resource Management) for representing language data and annotations. [...] The development of LAF and the ANC has been symbiotic: The ANC has effectively served as a testing ground for LAF, which has in turn evolved based on the experience gained in representing the ANC. This evolution has meant that the representation format for the ANC has changed from release to release, but now that LAF is stable, the ANC provides the first example of the state-of-the-art in language resource representation. (Ide 2009, S. 112).

Da die Spezifikation zum Zeitpunkt der Erstellung dieser Arbeit weiterhin nur als FDIS vorliegt, ist unklar, wann mit einer solchen Fassung zu rechnen ist.[15] Allerdings sind die Änderungen zwischen der momentan verfügbaren Version des OANC und dem GrAF-XSD relativ gering und beschränken sich auf das Auslagern der Metadaten

[14] Die in Abschnitt 9.2 gezeigten Listings 9.2 und 9.3 sind an ISO/DIS 24612 angepasst.

[15] Entgegen der zitierten Aussage „but now that LAF is stable“ ist laut der Webseite der ISO die Spezifikation weiterhin im Status „40.99“, was für den Hauptstatus „Enquiry Stage“ und den Unterstatus „DIS approved for registration as FDIS“ bedeutet, vgl. `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37326`.

(Header Document), sowie die Umstrukturierung einiger Elemente und Umbenennung einiger Attribute. Da das Datenmodell in beiden Fällen das eines Graphen ist, ist darüber hinaus eine automatisierte Konvertierung möglich. So demonstrieren Ide und Bunt (2010) eine Abbildung eines bestehenden Annotationsformats auf GRAF.

Der „Manually Annotated Sub-Corpus“ (MASC), eine händisch überarbeitete und erweiterte Teilmenge des OANC im Umfang einer halben Million Wörtern, nutzt ebenfalls GRAF als Annotationsformat (Ide, Suderman und Simms 2010). Auch hier ist allerdings unklar, welche Version des LINGUISTIC ANNOTATION FRAMEWORK zur Auszeichnung herangezogen wurde.

Kountz *et al.* (2008) beschreiben ein Schema für unterspezifizierte Repräsentationen syntaktischer Annotationen, d. h., solche, die ambig (in mehrfacher Hinsicht) sein können. Hierzu verwenden Sie ebenfalls ein auf LAF bzw. GRAF basierendes Format, erweitern es aber in der Hinsicht, dass auch die Kanten explizit mit Annotationen gelabelt werden (im Gegensatz zu der Vorgehensweise von GRAF, die die Einbeziehung von Dummy-Knoten vorsieht, die dem in Dörre (1997) demonstrierten AND-nodes bzw. OR-nodes folgt). Darüber hinaus fügen die Autoren dem Datenmodel von LAF generische Einschränkungen (*Generic Constraints*) hinzu, um alternative Lesarten auch ohne Hinzufügen neuer Dummy-Knoten und Kanten (und der dadurch auftretenden Vermischung von syntaktischer Annotation bzw. deren Knoten und Kanten und technischer Hilfsknoten und -kanten) realisieren zu können. Zu diesem Zweck wird eine *Constraint List* erstellt, die wiederum aus *Structural Constraints*, *Labelling Constraints* und *Constraint Interdepencies* bestehen kann und alle Einschränkungen beinhaltet, die für die Anordnung und das Labeln der Knoten im Baum eines Dokuments gelten sollen. Mittels der Structural Constraints werden Beziehungen zwischen nicht-ambigen Knoten im Baum festgelegt (in Form von Kanten, die über die Bezeichner der Knoten auf diese verweisen). Ein solches Constraint besteht aus dem Tripel (N_G, N_F, t), wobei N_F der Wurzelknoten des Teilgraphen ist, der unterhalb des Graphen mit dem Wurzelknoten N_G eingefügt werden soll. t ist dabei einer von (zum damaligen Zeitpunkt drei) vordefinierten Kombinationstypen von N_F und N_G, namentlich „BelowAppropriately“, „BelowWithOptions“ oder „AttachmentOption“. Im ersten Fall kann der Teilgraph mit dem Wurzelknoten N_F durch eine der in der Grammatik definierten passenden Positionen unterhalb des durch N_G gewurzelten Graphen eingefügt werden (z. B. beim PP-Attachment, vgl. auch die Ausführungen in Abschnitt 15.1). Der Kombinationstypus „BelowWithOptions“ ist analog zum ersten Fall, nur dass die möglichen Positionen im Graphen für das Einhängen des Teilgraphen nicht durch die Grammatik, sondern direkt im Constraint angegeben werden. Der dritte Fall wiederum besagt, dass N_F nur direkt unterhalb von N_G eingefügt werden kann.

Labelling Constraints legen das Verhältnis von Labeln zu Knoten und Kanten fest. Diese bestehen aus einem Tripel (S, L, t) mit S als Strukturelement, das durch das Label L näher spezifiziert werden soll. t dient der Interpretation von L – entweder als „LabelSet“ oder als „LabelSubtype“. Ist t ein LabelSet, wird L als Liste von möglichen Labeln für S aufgefasst, von deren Elementen nur eines zugewiesen werden kann. Im Falle eines LabelSubtype wäre L ein Supertyp von möglichen Labeln, die S erhalten kann, das konkrete Label wäre dann ein terminaler Subtyp der in L angegebenen Labelklasse. Hierzu wird das Tripel zu einem Quadrupel (S, F, L, t) erweitert, wobei F den Labeltyp

bezeichnet, der weiter eingeschränkt wird. Wichtig ist, dass das Strukturelement *S* sowohl Knoten als auch Kante im Graphen sein kann – und zwar ein(e) bereits bestehende(r) als auch ein(e) aus dem Einfügen eines Constraints resultierende(r). Im letzteren Fall mangelt es an einem feststehenden Bezeichner als Ankerpunkt, so dass Labelling Constraints auf das ursächliche Structural Constraint verweisen müssen, durch dessen Instanziierung das entsprechende Strukturelement erzeugt wird. Über die Constraint Interdependencies können Abhängigkeiten zwischen Constraint-Instanziierungen („Enforce“ bzw. „Preclude“) festgelegt werden, was die Kombination der Constraints ermöglicht.

Neben der Erweiterung des LAF-Datenmodells macht der Artikel Vorschläge für entsprechende neue Elemente (z. B. in Form des neu eingeführten `constraint-list`), um die GRAF-Notation anzupassen. Letztere sind aufgrund der vorangeschrittenen Entwicklung der Spezifikation nicht kompatibel, so dass abzuwarten ist, in wieweit neuere Versionen den aktuellen Stand berücksichtigen. Insgesamt zeigen Kountz *et al.* (2008) aber anschaulich die Anwendungs- und Erweiterungsmöglichkeiten von LAF, verdeutlichen aber auch Ungenauigkeiten im Standard (bezüglich der Möglichkeit Kanten zu annotieren).

Hayashi *et al.* (2010) nutzen ein LAF-/GRAF-basiertes Annotationsschema um multilinguale Dependenzstrukturen zu repräsentieren. Dabei werden – ähnlich wie im „Sekimo“-Projekt – verschiedene Annotationen als Ergebnis der Anwendung einer linguistischen Ressource zusammengefasst; hier allerdings mit multi-lingualen Daten. Zweck ist die Vergleichbarkeit von Dependenzstrukturen bzw. der leichtere Austausch solcher in Unabhängigkeit von den proprietären Formaten der Parser. Zu diesem Zweck werden neue Subkategorisierungsmechanismen dem Datenmodell hinzugefügt. Da diese allerdings weiterhin in Form von Merkmalsstrukturen in der GRAF-Notation gespeichert werden, muss hierzu nicht das Tagset erweitert werden (wie es bei Kountz *et al.* (2008) der Fall ist). Grundlage der Annotation ist das in Abschnitt 9.2.1 genannte RELAX-NG-Schema, das von dem im Anhang von ISO/DIS 24612 abweicht, da es noch Annotationen unterhalb von Kanten erlaubt (das Element `as` darf als Kind von `edge` auftreten, vgl. Abbildungen 7 und 9b in Hayashi *et al.* 2010). Der Ansatz, das Dump Format GRAF als Wrapper bzw. als Austauschformat für unterschiedliche linguistische Ressourcen nutzbar zu machen, ist dennoch interessant, wenn auch zum Zeitpunkt des Artikels die Konvertierungsarbeiten über ein händisch erstelltes Transformationsskript erfolgen müssen.

Die bisher geschilderten Anwendungsfälle machen deutlich, dass bereits vor Verabschiedung der finalen Version des LINGUISTIC ANNOTATION FRAMEWORK großes Interesse an der Spezifikation besteht. Das spiegelt sich auch darin wieder, dass für GRAF aufgrund seiner relativ simplen aber dennoch mächtigen Strukturierung bereits entsprechende Arbeiten unternommen wurden, um die Interoperabilität mit anderen Werkzeugen und Formaten aus der linguistischen Community sicherzustellen, namentlich GATE (GENERAL ARCHITECTURE FOR TEXT ENGINEERING, vgl. Cunningham 2002) und das bereits erwähnte UIMA (als CAS, *Common Analysis System Format*). So beschreiben Ide und Suderman (2010) einen *Round Trip* zwischen den genannten drei Formaten. Die dazu notwendige Software in Form von Konvertierungswerkzeugen und Plugins für

das GATE-System sind auf der Homepage des ANC frei verfügbar.[16] Es ist allerdings unklar, inwieweit diese Programme zum aktuellen Stand der Spezifikation kompatibel sind bzw. laufend aktualisiert werden.

9.4 Aktueller Stand und Beurteilung

Das LINGUISTIC ANNOTATION FRAMEWORK ist eine interessante Spezifikation, die bereits jetzt einen breiten Zuspruch in der linguistischen Community erfahren hat. Zu bewerten sind hierbei sowohl das Datenmodell in Form eines gerichteten Graphen als auch die propagierte Notation in Form des Dump Formats GRAF.[17] Zum Datenmodell gelten die bereits in Abschnitt 3.3.3 gemachten Aussagen: Ein Graph eignet sich prinzipiell besser als ein gerichteter Baum zur Darstellung linguistischer Strukturen, da Relationen zwischen den Knoten weniger eingeschränkt werden und damit u. a. auch diskontinuierliche Annotationen möglich sind. In diesem Sinne wird der Standard bereits extensiv genutzt, ohne dass eine finale Fassung vorliegt. Das Datenmodell betreffend ist der Einsatz auch relativ unproblematisch, da in der Nutzerschaft Übereinstimmung herrscht mit Graphen als formalen Modellen zu arbeiten (auch bei alternativen Ansätzen wie dem in Teil III diskutierten XSTANDOFF). Dazu kommt, dass dieser Teil der Spezifikation durch mehrere Versionen hinweg stabil geblieben ist, was sicherlich auch der längeren Entstehungsgeschichte (über die Stationen TEI GUIDELINES und CORPUS ENCODING STANDARD) geschuldet ist.

Ebenso positiv beurteilt wird das ursprünglich als reines Dump Format konzipierte GRAF. Problematisch sind hierbei zwei Punkte: Zum einen ist das XML-Schema selbst in ISO/DIS 24612 nur im informativen Teil der Spezifikation enthalten, was die Positionierung als Austauschformat (und damit als Ausgangs- und Zielpunkt für Transformationen) beeinträchtigt. Zum anderen ist gerade dieser Teil des Standards häufig überarbeitet worden, angefangen von der Änderung des Zielnamensraums bis hin zu der unklaren Haltung gegenüber Kanten zugewiesenen Annotationen, die in der oft zitierten und letzten frei verfügbaren Fassung (ISO/CD 24612) noch unterstützt werden, in ISO/DIS 24612 allerdings wieder fehlen und in ISO/FDIS 24612 erneut enthalten sind. Gerade dieser letzte Punkt gibt Anlass zu Irritationen, da selbst aktuelle Publikationen wie Hayashi *et al.* (2010) diese Möglichkeit nutzen. Darüber hinaus steht mit dem in Hayashi *et al.* (2010) zitierten RELAX-NG-Schema eine weitere Dokumentgrammatik zur Verfügung, so dass insgesamt drei zueinander nicht vollständig kompatible Fassungen im Umlauf sind.

Von einem technischen Standpunkt her ist das in ISO/DIS 24612 enthaltene XSD als eher schlicht einzustufen: Die Möglichkeiten, die XML SCHEMA bietet und die in Abschnitt 3.4 diskutiert werden, werden nicht ansatzweise genutzt. Gemessen an der Ausdrucksstärke bleiben die in GRAF genutzten Merkmalsstrukturen auf dem Stand einer lokalen Baumgrammatik und lassen sich damit auch mittels XML-DTD spezifizie-

[16] Weitere Informationen unter `http://www.anc.org`, zuletzt abgerufen am 19.04.2012.

[17] Zur Abgrenzung des Datenmodells von LAF zu dem des ANNOTATION GRAPH vgl. Wörner (2009, S. 104;106f.).

ren.[18] Der Verzicht auf Integritätsmerkmale (die allerdings aufgrund der Aufteilung in verschiedene Dateien auch nicht sinnvoll einsetzbar wären) erlaubt prinzipiell auch die Nutzung von RELAX NG als Constraint Language, würde aber dessen Möglichkeiten bei Weitem nicht ausschöpfen. Allerdings würde es die Interoperabilität von LAF/GRAF mit den weiteren in ISO/TC 37/SC 4 entwickelten Normen erleichtern, von denen LAF referenziert wird (beispielsweie das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK, vgl. Kapitel 10).

Ende August 2011 wurde die Spezifikation nach längerer Überarbeitung aufgrund entsprechender Kommentare der stimmberechtigten P-Mitglieder im Status eines FDIS veröffentlicht.. Diese ebenfalls nicht kostenfrei verfügbare Version adressiert einige der angeführten Kritikpunkte, unterscheidet sich aber dementsprechend noch stärker von den Vorversionen. Das Datenmodell wurde konkretisiert, indem Ankerpunkte zur Definition von Abschnitten und die Graphenstruktur explizit aufgeführt werden, zum Vergleich seien die entsprechenden Definitionen aus ISO/DIS 24612 und ISO/FDIS 24612 herangezogen.

> The LAF data model consists of (1) a *referential structure* for associating stand-off annotations with primary data; and (2) a *feature structure representation* for annotation content. The data model for annotations thus comprises a directed graph referencing n-dimensional regions of primary data as well as other annotations, in which nodes are labeled with feature structures providing the annotation content. (ISO/DIS 24612, S. 5).

> The LAF data model consists of (1) a structure for describing media, consisting of *anchors* that reference locations in primary data and *regions* defined in terms of these anchors; (2) a *graph structure*, consisting of nodes, edges, and links to regions; and (3) an *annotation structure* for representing annotation content with feature structures. The data model for annotations thus comprises a directed graph referencing n-dimensional regions of primary data as well as other annotations, in which nodes are associated with feature structures providing the annotation content. (ISO/FDIS 24612, S. 7f.).

Eine grafische Übersicht ist in Abbildung 9.3 aus ISO/FDIS 24612, S. 8 dargestellt.

Entfernt wurde die Dokumentgrammatik für GRAF. Dafür wurden sowohl RELAX NG-Fragmente als auch exemplarische (und nicht immer wohlgeformte) XML-Instanzen zur Dokumentation des Aufbaus des Austauschformats hinzugefügt. Demnach gehört das Wurzelelement `graph` dem Namensraum `http://www.xces.org/ns/GrAF/1.0/` an und besteht aus einem obligatorischen `graphHeader` (dem Header Document für die Annotation), sowie optional Knoten (`node`), Kanten (`edge`), Annotationen (`a` [sic!]) und Ankern (`anchor`). In dieser Fassung ist die Zuweisung von Annotationen an Kanten wieder möglich, so dass dieser Mangel behoben wurde. Zusätzlich wurde mit den Elementen `roots` bzw. `root` eine explizite Auszeichnung von Wurzelknoten in Graphen vorgesehen, die eine Baum- oder Waldstruktur haben (die Zuordnung erfolgt über das `node.id`-Attribut). Es ist allerdings unklar, an welcher Position in der Instanz

[18] Mit der Einschränkung der fehlenden Unterstützung von XML NAMESPACES.

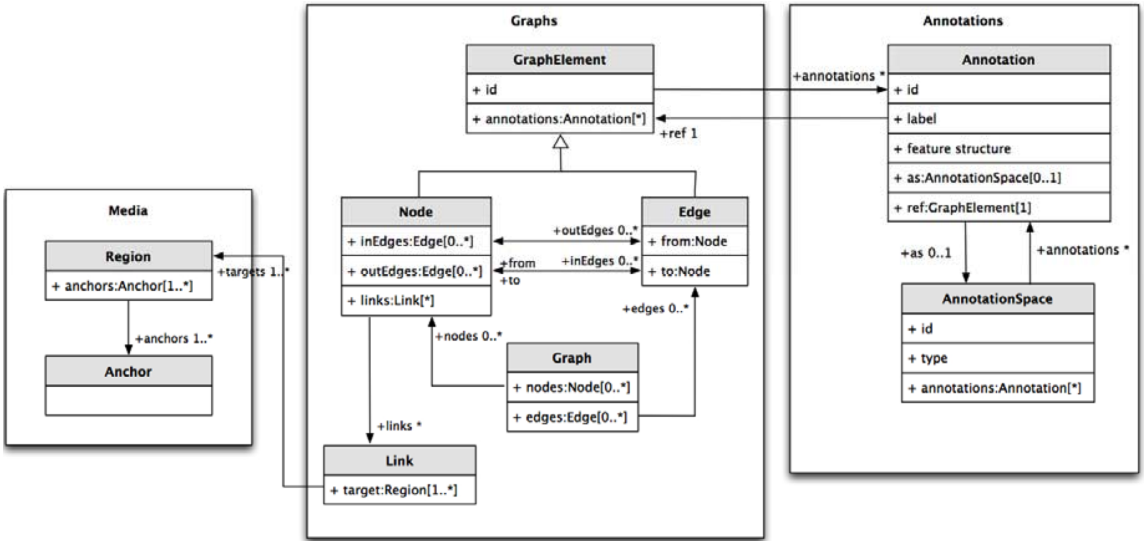


Abbildung 9.3: Das aktualisierte Datenmodell

die beiden Elemente auftreten können, da sie zwar in der tabellarischen Aufstellung enthalten sind, nicht aber in der RELAX NG-ähnlichen Darstellung der Inhaltsmodelle in ISO/FDIS 24612, S. 14.[19] Änderungen betreffen auch die Metadaten für die Primärdaten (*Primary Data Document Header*, Element `documentHeader`) und die Annotationen (*Annotation Document Header*, Element `graphHeader`). Listing 9.4 zeigt eine Beispiel-Instanz (entnommen aus ISO/FDIS 24612, S. 17).[20]

Die Darstellung des eigentlichen Graphen beginnt in Zeile 19. Der hier gezeigte Knoten wird über das `xml:id`-Attribut eindeutig bezeichnet. Über diesen Bezeichner wird in der nächsten Zeile der Knoten mit einer Annotation versehen, die ähnlich (allerdings nicht identisch) aufgebaut ist, wie die in Kapitel 6 diskutierten Merkmalsstrukturbeschreibungen. In Zeile 29 wird eine Kante zwischen dem Knoten und einem weiteren (im Listing nicht enthaltenen Knoten) hergestellt, während Zeile 31 die Definition eines Bereichs in den Primärdaten demonstriert, der durch die Anker mit den Werten *980* und *9190* begrenzt wird.[21] Die Interpretation der Zeichenketten des `anchors`-Attribut ist ISO/FDIS 24612, S. 15 folgend anwendungsabhängig: „The anchors attribute contains a whitespace-delimited list of values that represent the anchor values. Applications are expected to know how to parse the string representation of an anchor into a location in the artifact being annotated.“ Dagegen ist der Knoten in Zeile 34 durch ein `link`-Element näher definiert, das über sein `targets`-Attribut auf die Region in Zeile 31 verweist – womit dieser Knoten im Graphen (und die damit verknüpften Annotationen)

[19] Sinnvoll wäre eine Positionierung auf der Ebene der Elemente `node`, `edge`, `a` und `anchor`.

[20] Die hier gezeigte Version wurde gegenüber der Originalinstanz abgeändert, um Wohlgeformtheit herzustellen.

[21] Das Element `region` wird – analog zu den Elementen `root` und `roots` nur in der tabellarischen Aufstellung in ISO/FDIS 24612, S. 15 aufgeführt, fehlt aber in der RELAX NG-ähnlichen Darstellung des Inhaltsmodells in ISO/FDIS 24612, S. 14. Es ist anzunehmen, dass es – wie aus der Beispiel-Instanz ersichtlich – als Geschwisterknoten der Elemente `node`, `edge`, `a` und `anchor` auftreten kann.

Listing 9.4: GrAF-Instanz mit Metadaten und Annotationsfragmenten

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <graph xmlns="http://www.xces.org/ns/GrAF/1.0/">
3   <graphHeader>
4    <labelsDecl>
5     <labelUsage label="fullTextAnnotation" occurs="1"/>
6     <labelUsage label="Target" occurs="171"/>
7     <labelUsage label="FE" occurs="372"/>
8     <labelUsage label="sentence" occurs="32"/>
9     <labelUsage label="annotationSet" occurs="171"/>
10    <labelUsage label="NamedEntity" occurs="32"/>
11   </labelsDecl>
12   <dependencies>
13    <dependsOn type="fntok"/>
14   </dependencies>
15   <annotationSpaces>
16    <annotationSpace as.id="FrameNet" default="true"/>
17   </annotationSpaces>
18  </graphHeader>
19  <node xml:id="fn-n156"/>
20  <a label="FE" ref="fn-n156">
21   <fs>
22    <f name="FE" value="Speaker"/>
23    <f name="rank" value="1"/>
24    <f name="GF" value="Ext"/>
25    <f name="PT" value="NP"/>
26   </fs>
27  </a>
28  <!-- [...] -->
29  <edge xml:id="e233" from="fn-n156" to="fn-n133"/>
30  <!-- [...] -->
31  <region xml:id="r1" anchors="980 9190"/>
32  <region xml:id="r2" anchors="980 993"/>
33  <!-- [...] -->
34  <node xml:id="a232">
35   <link targets="r1"/>
36  </node>
37  <node xml:id="a233">
38   <link targets="r2"/>
39  </node>
40  <!-- [...] -->
41  <a label="R Gesture Units 1" ref="a232"/>
42  <a label="preparation" ref="a233"/>
43 </graph>
```

sich direkt auf die durch diese Region begrenzten Primärdaten bezieht. Wie in Zeile 41 einsehbar, können a-Elemente auch leer sein und nur dazu genutzt werden, um andere Knoten mit einem Label zu versehen.

Abschließend zeigt das Beispiel in Listing 9.4 die starke Überarbeitung, die beim LINGUISTIC ANNOTATION FRAMEWORK in der vorletzten Phase der Normierung stattgefunden hat. Positiv ist daran hervorzuheben, dass diese aktuelle Fassung der Spezifikation sich auf die Definition des formalen Modells beschränkt, was die Nutzung in Kombination mit anderen Standards erleichtert. Zusätzlich hat eine Reihe von Überarbeitungen stattgefunden, die Anregungen anderer P-Mitglieder aufgegriffen haben. Die Verknüpfungen zu der in Kapitel 6 diskutierten standardisierten Beschreibung von Merkmalsstrukturen sind deutlicher herausgearbeitet. Auch sind die Ausführungen

zu GRAF merklich fundierter und klarer strukturiert, sowohl den Aufbau als auch die Metadaten betreffend. Ebenfalls sinnvoll ist die Nutzung eines neuen Namensraumes, so dass nach ISO/FDIS 24612 modellierte Instanzen einfach von früheren Fassungen unterschieden werden können.

Als negativ kann allerdings die Entfernung der Dokumentgrammatik für das GRAPH ANNOTATION FORMAT angesehen werden, da es als Austauschformat weiterhin Teil der Spezifikation ist. Ebenso kritikwürdig ist die Tatsache, dass für die Strukturierung der Merkmalsstrukturen nicht vollständig die entsprechende Norm ISO 24610-1:2006 (vgl. Kapitel 6) genutzt wird. So wird das Element `f` in ISO/FDIS 24612, S. 16 wie folgt beschrieben:

> An attribute/value pair. In the concise form (given here), the <f> element is empty and includes attributes providing simple name/value pairs. More complex feature structures may be represented according to the specification in ISO 24610-1, Language resource management – Feature structures – Part 1: Feature structure representation (FSR), which should be consulted for details.

Zwar wird auf ISO 24610-1:2006 verwiesen, allerdings nur als Alternative zur kürzeren Darstellung bzw. zur Kodierung komplexer Merkmalsstrukturen. Ebenso ist nicht ganz nachvollziehbar, warum die Verknüpfung von Knoten (Element `node`) und Annotation (Element `a`) über den Bezeichner (und damit über eine `ID`/`IDREF`-Relation) hergestellt wird, obwohl auch denkbar gewesen wäre, das `a`-Element neben `link` als Kind des `node`-Elements (und des `edge`-Elements) zuzulassen. Diese Fragen können aber noch durchaus Gegenstand der weiteren Normierungsarbeit sein.

Da das LINGUISTIC ANNOTATION FRAMEWORK sein Annotationsinventarium durch Definition einer DCS bezieht, lassen sich – wie bereits in Abschnitt 8.3 angesprochen – konzeptuelle Ebene und Notation klar voneinander trennen.

Zusammenfassend bleibt festzuhalten, dass die Spezifikation trotz der jahrelangen Entwicklung weiterhin als noch nicht abgeschlossen einzustufen ist – mit den entsprechenden Konsequenzen für Projekte und Korpora, die auf eine bisherige Fassung des Standards gesetzt haben. Allerdings kann damit gerechnet werden, dass eine endgültige Verabschiedung als internationale Norm in nicht allzu weiter Zukunft erfolgen wird.

10 Morpho-syntactic Annotation Framework (MAF)

Wie bereits in den Ausführungen zu den TEI GUIDELINES (Kapitel 5) und dem CORPUS ENCODING STANDARD (Kapitel 7) angemerkt, ist ein standardisiertes Annotationsinventar zur Auszeichnung speziell syntaktischer Merkmale nicht vorhanden – die TEI klammert diesen Aspekt aus und die aktuelle Fassung XCES sowie das LINGUISTIC ANNOTATION FRAMEWORK (Kapitel 9) setzen auf generische Merkmalsstrukturen. Dabei werden erste Bemühungen für ein generalisiertes Framework zur Syntax-Annotation auf Basis von XCES und einer Datenkategorienregistrierung bereits in Ide, Romary und Erjavec (2001) skizziert, die dann in entsprechende Normierungsbemühungen im Rahmen von ISO/TC 37/SC 4 mündeten. In der Arbeitsgruppe 2 (WG 2, „Representation schemes“) werden seit einigen Jahren die internationalen Normen MORPHO-SYNTACTIC ANNOTATION FRAMEWORK (MAF) und SYNTACTIC ANNOTATION FRAMEWORK (SYNAF, vgl. Kapitel 11) vorbereitet. Während SYNAF bereits als Internationaler Standard ISO 24615:2010 verabschiedet wurde, ist MAF seit 2008 im Status DIS (ISO/DIS 24611), die letzte frei verfügbare Fassung ist hier der Committee Draft aus dem Jahr 2005 (ISO/CD 24611). Eine zusätzlich frei verfügbare Beschreibung ist in Clément und de la Clergerie (2005) zu finden.

10.1 Aufbau

Das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK definiert ein Metamodell zur Repräsentation morphosyntaktischer Annotationen. Zusätzlich enthält der Standard im informativen Anhang ein RELAX-NG-Schema und eine Datenkategorieauswahl (Data Category Sets, DCS, vgl. Kapitel 8). Dem Datenmodell folgend besteht ein annotiertes Dokument aus den Primärdaten (*Raw Document*) und den eigentlichen Annotationen (im Standoff-Verfahren). Die Annotationsebene besteht aus Wortformen, die über Zeichenspannen der Primärdaten begrenzt werden. Eine Wortform kann dabei einen Bereich

über kein Segment oder mehrere Segmente umfassen. Darüber hinaus enthält eine Wortform Information über das Lemma und optionale Verweise auf Lexikoneinträge – hier wäre ein logischer Anknüpfungspunkt an das LEXICAL MARKUP FRAMEWORK (LMF, ISO 24613:2008), der allerdings in der vorliegenden Fassung der Spezifikation fehlt. Morphosyntaktische Informationen der Wortform werden mittels Merkmalsstrukturbeschreibungen gemäß ISO 24610-1:2006 annotiert (vgl. Kapitel 6), wobei die Kategorien in einer Datenkategorienregistrierung gemäß ISO 12620:2009 gespeichert werden (vgl. Kapitel 8). Die Segmentierung erfolgt mittels Token, die – ebenso wie Wortformen – in gerichteten azyklischen Graphen (DAG) organisiert werden können, die hier als Tokennetz bzw. Wortformennetz (wörtlich: *Token Lattice* und *Word Form Lattice*) bezeichnet werden (vgl. Abschnit 10.1.3).

Abbildung 10.1 zeigt eine deutsche Fassung des vereinfachten Metamodells aus ISO/DIS 24611. Die Instanziierung eines MAF-Modells erfolgt durch Auswahl eines Data Category Sets (ISO/DIS 24611, S. 8).

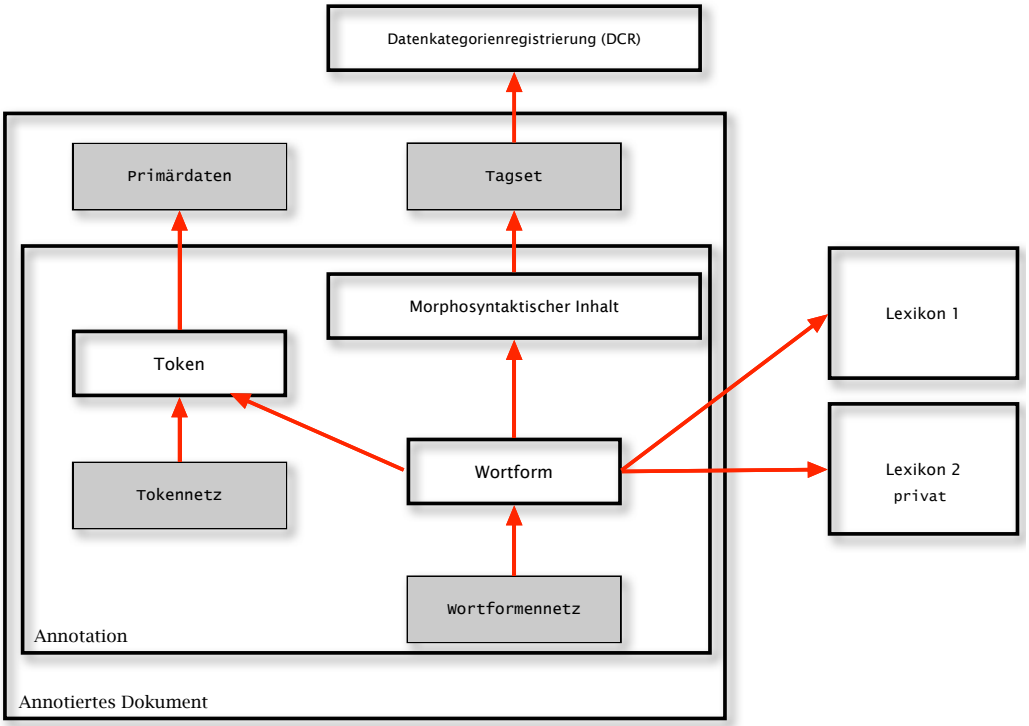


Abbildung 10.1: MAF-Metamodell

Token dienen als Anker für spätere Annotationen und definieren nicht-leere, fortlaufende Zeichenkettensegmente in den Primärdaten, die beispielsweise das Resultat einer Verarbeitung des Originaldokuments mittels Tokenizer sein können (Clément und de la Clergerie 2005, S. 91). Es ist daher zu unterscheiden zwischen den morphosyntaktischen Annotationen, deren Träger die Token sind, und den enthaltenen Wortformen – beispielsweise bei Komposita (ISO/DIS 24611, S. 13).[1] Die Abgrenzung

[1] Zu beachten ist der Unterschied, dass innerhalb der Token die Zeichenketten fortlaufend, d. h., in der

von Token zueinander kann unterschiedlich motiviert sein, die Spezifikation sieht hier je nach Sprache typographische, morphologische oder andere Grenzen vor (ISO/DIS 24611, S. 13). Im Skopus des Standards sind nur solche Token relevant, die Träger morphosyntaktischer Annotationen sind, Absatz- oder Seitengrenzen o. ä. Konstrukte werden nicht beachtet, daher sind nicht alle Zeichen der Primärdaten zwingend mittels Token ausgezeichnet.

Eine Beispiel-Instanz mit dem Primärdatum „I wanna put up new wallpaper." zeigt Listing 10.1.[2] Hier handelt es sich um eine Inline-Notation, d. h., Token und Primärdaten sind in derselben Datei gespeichert (in diesem Fall ist die Reihenfolge der `token`-Elemente natürlich durch die Zeichenfolge in den Primärdaten bestimmt).

10.1.1 Tokenisierung

Die Token mit den Bezeichnern `t2` und `t3` (Zeilen 4 und 5, Zeichenkette „wanna" in den Primärdaten) werden mittels der beiden Wortformen in den Zeilen 19 und 26 („want" und „to") morphosyntaktisch annotiert. Dagegen werden die beiden Token `t4` und `t5` (Zeilen 6 und 7) in der Wortform in Zeile 36 („put up") zusammengefasst.

Listing 10.1: MAF-Beispiel-Instanz

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <maf>
3   <token id="t1">I</token>
4   <token id="t2" join="right">wan</token>
5   <token id="t3" join="left">na</token>
6   <token id="t4">put</token>
7   <token id="t5">up</token>
8   <token id="t6">new</token>
9   <token id="t7">wall</token>
10  <token id="t8">paper</token>
11  <token id="t9">.</token>
12  <wordForm lemma="I" tokens="t1">
13   <fs>
14    <f name="pos">
15     <symbol value="PP"/>
16    </f>
17   </fs>
18  </wordForm>
19  <wordForm lemma="want" tokens="t2">
20   <fs>
21    <f name="pos">
22     <symbol value="VBP"/>
23    </f>
24   </fs>
25  </wordForm>
26  <wordForm lemma="to" tokens="t3">
27   <fs>
28    <f name="pos">
29     <symbol value="TO"/>
```

korrekten Reihenfolge vorliegen, während die Token als solche nicht zwingend in einer festgelegten Weise in der Annotation auftauchen müssen – je nachdem, ob Inline- oder Standoff-Annotation verwendet wird.

[2] Die Instanz wurde von der Webseite `http://korpling.german.hu-berlin.de/tiger2/homepage/tiger1.html` entnommen und adaptiert, um erfolgreich gegen die vorliegende MAF-Dokumentgrammatik validieren zu können. Zuletzt abgerufen am 19.04.2012.

```
30    </f>
31   </fs>
32  </wordForm>
33  <wordForm tokens="t2 t3"/>
34  <wordForm lemma="put" tokens="t4"/>
35  <wordForm lemma="up" tokens="t5"/>
36  <wordForm lemma="put_up" tokens="t4 t5">
37   <fs>
38    <f name="pos">
39     <symbol value="VB"/>
40    </f>
41   </fs>
42  </wordForm>
43  <wordForm lemma="new" tokens="t6">
44   <fs>
45    <f name="pos">
46     <symbol value="JJ"/>
47    </f>
48   </fs>
49  </wordForm>
50  <wordForm lemma="wallpaper" tokens="t7 t8">
51   <fs>
52    <f name="pos">
53     <symbol value="NN"/>
54    </f>
55   </fs>
56  </wordForm>
57 </maf>
```

Die hier demonstrierte Inline-Annotation wird im Standard beschrieben und für einfache Beispiele herangezogen, es wird aber auch darauf hingewiesen, dass sie nicht die empfohlene Herangehensweise repräsentiert:

> The embedding notation will be used for most of the provided examples for MAF but it should be noted that the use of this notation is not recommended. A first reason is that the morpho-syntactic annotations may conflict with other annotations. A second reason is that the content of the textual material separating the textual content embedded within `<token>` is not precisely defined (white-space, newlines, no space, hyphen, ...), except by relying on attribute join. (ISO/DIS 24611, S. 14)

Vielmehr wird das Standoff-Verfahren präferiert. Hierbei können die `token`-Elemente über die Attribute `from` und `to` Intervalle in den Primärdaten definieren, die Gegenstand morphosyntaktischer Annotation sein sollen. Die Spezifikation erwähnt hier keine konkreten Segmentierungsmechanismen: „The content of these attributes depends on some chosen addressing schema to denote non ambiguous document positions and depends on the nature of the original document.“ (ISO/DIS 24611, S. 14). An anderer Stelle wird auf das LINGUISTIC ANNOTATION FRAMEWORK verwiesen, dessen Spezifikation mögliche Adressierungsschemata in Form von Werten des Attributs `addressing` des Wurzelelements `maf` enthalten soll: „The addressing attribute indicates the addressing schema used to refer positions in the annotated document. A full list of such schema will be provided in ISO 24612 proposal ‚Linguistic Annotation Framework‘ (LAF).“ (ISO/DIS 24611, S. 35) Die unter der URL `http://atoll.inria.fr/~clerger/MAF/maf.rnc` zu findende Dokumentgrammatik `maf.rnc` nutzt hierzu den Datentyp `xs:NMTOKEN` und

Tabelle 10.1: Tokengrenzen auf Basis der Zeichenpositionen

Wortform	Startposition	Endposition
I	0	1
wan	2	5
na	5	7
put	8	11
up	12	14
new	15	18
wall	19	23
paper	23	28
.	28	29

verweist auf das LINGUISTIC ANNOTATION FRAMEWORK (vgl. Kapitel 9).[3] In Clément und de la Clergerie (2005, S. 91) werden als mögliche Mechanismen für die hier *Document Positions* genannten Angaben u. a. Byte Offsets, Zeichenpositionen oder Zeitangaben angegeben. Würde man Zeichenpositionen heranziehen, ergäbe sich die in Tabelle 10.1 dargestellte Aufteilung für die Wortformen.

Dieser Form der Segmentierung folgend, könnten die `token`-Elemente aus Listing 10.1 durch die in Listing 10.2 ersetzt werden. In diesem Fall kann über das `addressing`-Attribut des Wurzelelements eine Angabe zum verwendeten Adressierungsschema der Positionen in den Primärdaten gemacht werden, die zur Definition der Token herangezogen werden (der im Listing gezeigte Wert *char_offset* ist ISO/DIS 24611, S. 35 entnommen). Das Attribut `document` dient der Referenz auf die Primärdatendatei.

Listing 10.2: Token-Elemente in Standoff-Notation

```
<maf document="sample.txt" addressing="char_offset">
 <token id="t1" form="I" from="0" to="1"/>
 <token id="t2" join="right" form="wan" from="2" to="5"/>
 <token id="t3" join="left" form="na" from="5" to="7"/>
 <token id="t4" form="put" from="8" to="11"/>
 <token id="t5" form="up" from="12" to="14"/>
 <token id="t6" form="new" from="15" to="18"/>
 <token id="t7" form="wall" from="19" to="23"/>
 <token id="t8" form="paper" from="23" to="28"/>
 <token id="t9" form="." from="28" to="29">.</token>
 <!-- [...] -->
</maf>
```

Die bereits in Abschnitt 3.3.2.3 dieser Arbeit diskutierten Vorteile einer Standoff-Annotation bestehen darin, dass auch die Verwendung mehrerer Token, die dieselbe Textspanne (oder überlappende Textspannen) markieren, zulässig ist (beispielsweise bei Agglutinationen). Zusätzlich können im ersten Fall auch zwei Wortformen sich auf ein und dasselbe Token beziehen, so dass sich die Überlappung auf dieser Ebene manifestiert (ISO/DIS 24611, S. 16). Zu diesen Zwecken kann das optionale `join`-Attribut des `token`-Elements die Werte *no* (Standardwert), *left*, *right*, *both* und *overlap*

[3] Es sollte beachtet werden, dass dieses RNG-Schema laut Kommentar aus dem September 2005 stammt, also nicht mehr der aktuellen Version der Spezifikation entsprechen dürfte. Letzter Abruf der Datei über die genannte URL am 19.04.2012. Das im informativen Anhang von ISO/DIS 24611 enthaltene RNG-Schema definiert beide Attribute als leer (`empty`). Allerdings enthält diese Dokumentgrammatik einige Fehler, so dass die Ausführungen sich vorrangig auf das syntaktische korrekte `maf.rnc` beziehen.

annehmen, wobei letzterer der Auszeichnung überlappender Segmente dient. Die Werte *left*, *right* und *both* zeigen an, dass das entsprechende Token mit seinem linken, rechten bzw. beiden Geschwistern zusammenhängt (im Beispiel die Token `t2` in Zeile 4 und `t3` in Zeile 5, die gemeinsam die Zeichenkette „wanna“ bilden). Dieses Attribut sollte vorrangig in der Inline-Annotation eingesetzt werden, um die gegenüber dem Standoff-Verfahren geringere Präzision bzgl. der Segmentgrenzen (z. B. durch Weißraum-Zeichen) zu verbessern. Die Spezifikation führt darüber hinaus Beispiele für die Relation von Token zu Wortform auf, um die Behandlung von Komposita oder diskontinuierlichen Einheiten zu demonstrieren (vgl. Abschnitt 7.1, „Token attachment“ in ISO/DIS 24611, S. 18f. sowie Clément und de la Clergerie 2005, S. 91). Über die optionalen Attribute `form`, `phonetic`, `transcription` und `transliteration` können zusätzliche Angaben über eine korrekte Form (beim Vorliegen von Schreibfehlern), phonetische Transkription und Angaben zu alternativen Schreibweisen gemacht werden.

10.1.2 Wortformen

Eine Wortform im Sinne des MORPHO-SYNTACTIC ANNOTATION FRAMEWORK ist eine linguistische Einheit, die durch ihre morphosyntaktischen Eigenschaften identifiziert wird (Clément und de la Clergerie 2005, S. 91) und in der XML-Notation durch das `wordForm`-Element ausgezeichnet wird. Dieses kann durch eine Reihe von optionalen Attributen weiter spezifiziert werden (vgl. auch Zeile 19 in Listing 10.1 auf Seite 217), darunter Angaben zum Lemma, der Form (Attribut `form`) oder den Token-Ankern – wobei der Bezug zu letzteren durch den `xs:IDREFS`-Datentyp, also ein Integritätsmerkmal hergestellt wird. Durch das Attribut `entry` können optional Bezüge zu Lexikoneinträgen gemacht werden, die die Wortform näher erläutern: „In particular, the use of document specific ‚lexica‘ was suggested, for collecting and organizing the named entities found in a document.“ (Clément und de la Clergerie 2005, S. 90). Da diese allerdings außerhalb des Skopus' der Spezifikation liegen, werden hierzu keine näheren Angaben gemacht (vorgeschlagen werden *Uniform Resource Names*, URN; die Dokumentgrammatik verwendet den Datentyp `xs:anyURI`).

Unterhalb eines `wordForm`-Elements werden die bereits aus Kapitel 6 bekannten standardisierten Merkmalsbeschreibungen genutzt, um die eigentlichen morphosyntaktischen Angaben strukturiert zu speichern (in der Abbildung 10.2 auf Seite 222 als „Morphosyntaktischer Inhalt“ bzw. „Tagset“ bezeichnet, wobei letzteres dem Layer, ersteres dem Level aus Abschnitt 3.5 entspricht). Dabei sollen als Merkmale und deren Werte Konzepte aus einer Datenkategorienregistrierung gemäß ISO 12620:2009, beispielsweise das in Abschnitt 8.2 diskutierte ISOCAT, zum Einsatz kommen. Alternativ (oder ergänzend) ist die Definition eines eigenen *Tagsets* möglich. Ein solches wird durch das gleichnamige Element festgelegt, das unterhalb des `maf`-Wurzelements auftauchen darf (also ein Geschwisterknoten der Elemente `token` und `wordForm` ist). Listing 10.3 zeigt ein abgewandeltes Beispiel aus ISO/DIS 24611, S. 24f. Das `tagset`-Element kann beliebig viele Vorkommen der Elemente `dcs`, `fsd`, `fvLib` und `fLib` beinhalten, wobei die letzten drei ISO 24610-1:2006 entnommen sind und bereits in Kapitel 6 vorgestellt wurden. Das Element `dcs` leitet die Auswahl eines Data Category Sets ein. Über das `local`-Attribut wird ein lokaler Name vergeben, der Wert des

registered-Attributs verweist auf eine externe Datenkategorienregistrierung, das rel-Attribut kann genutzt werden, um die Beziehung zwischen beiden DCS zu spezifizieren. So ist das in Zeile 4 definierte DCS eine Teilmenge des externen DCS, das über den Bezeichner dcs:morphosyntax:pos:adverb identifiziert wird, während die beiden anderen äquivalent sind.

Listing 10.3: Definition eines Tagsets

```
1 <tagset>
2  <dcs local="genre" registered="dcs:morphosyntax:gender:fr" rel="eq"/>
3  <dcs local="fem" registered="dcs:morphosyntax:gender:fr:feminine" rel="eq"/>
4  <dcs local="advneg" registered="dcs:morphosyntax:pos:adverb" rel="subs"/>
5 </tagset>
```

Neben der in Listing 10.1 auf Seite 217 gezeigten üblichen Darstellung der Merkmalsstrukturen per fs-Element ist auch eine kompakte Variante möglich. In dieser ist das wordForm-Element leer und enthält die durch Leerzeichen getrennten Merkmal-Wert-Paare (in sich getrennt durch einen „.“) als Wert des tag-Attributs. Listing 10.4 zeigt ein Beispielfragment (entnommen aus ISO/DIS 24611, S. 22).

Listing 10.4: Fragment einer Beispielinstanz in kompakter Schreibweise

```
1 <token id="t0">belle</token>
2 <wordForm tokens="t0" entry="urn:lexicon:fr:beau" tag="pos.adj adj_type.qual gender.fem num.
      sing"/>
```

Die Nutzung von Feature Libraries bzw. Feature-Value Libraries (analog zu ISO 24610-1:2006) wird ebenfalls unterstützt. Hierzu können die bereits angesprochenen und in Kapitel 6 diskutierten Elemente fvLib und fLib eingesetzt werden.

10.1.3 Ambiguitäten

Abgesehen von den bereits aufgeführten Elementen tagset, token und wordForm können unterhalb des Wurzelelements maf noch das fsm-Element und das wfAlt stehen. Das Element wfAlt dient der Kennzeichnung lexikalischer Ambiguitäten und gruppiert entsprechende wordForm-Elemente, während ambige Wortformen (morphologische Ambiguitäten) durch das vAlt als alternative Werte des entsprechenden Merkmals kodiert werden (vgl. die Listings 10.5 und 10.6, die adaptierte Fassungen der in ISO/DIS 24611, S.27f. enthaltenen sind).

Listing 10.5: Morphologische Ambiguität

```
1  <token id="t0">mange</token>
2  <wordForm tokens="t0" entry="urn:lexicon:fr:manger">
3   <fs>
4    <f name="pos">
5     <symbol value="verb"/>
6    </f>
7    <f name="tense">
8     <symbol value="present"/>
9    </f>
10   <f name="person">
11    <vAlt>
12     <symbol value="first"/>
13     <symbol value="third"/>
14    </vAlt>
```

```
  </f>
  <f name="number">
   <symbol value="singular"/>
  </f>
 </fs>
</wordForm>
```

Listing 10.6: Lexikalische Ambiguität

```
<token id="t0">porte</token>
<wfAlt>
 <wordForm tokens="t0" entry="urn:lexicon:fr:porte" tag="pos.n"/>
 <wordForm tokens="t0" entry="urn:lexicon:fr:porter" tag="pos.v"/>
</wfAlt>
```

Zur Kodierung struktureller Ambiguitäten wird das `fsm`-Element eingesetzt. Der Elementname bezeichnet dabei eine Finite State Machine, also einen endlichen Automaten, der eingesetzt wird, um mögliche Lesarten in Form verschiedener Pfade zu kodieren.[4] Solche strukturellen Ambiguitäten werden in ISO/DIS 24611 *Lattice* genannt (in Abbildung 10.1 auf Seite 216 jeweils als „Netz“ tituliert): „Lattices may also be seen either as a restricted kind of *finite state automata* or as an extension of Directed Acyclic Graphs (DAGs).“ (Clément und de la Clergerie 2005, S. 90)

In ISO/TC 37/SC 4/WG 2 (ISO/DIS 24611, S. 28f.) wird als Beispiel das französische „fer à cheval“ (Hufeisen) angeführt, Abbildung 10.2 zeigt einen möglichen endlichen Automaten.

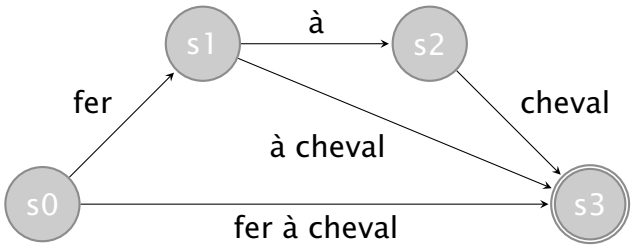


Abbildung 10.2: Möglicher FSA für die Phrase „fer à cheval“

Die Zerlegung in die Einheiten „fer à cheval“ (das Nomen „Hufeisen“), „fer“ (das Nomen „Eisen“), „à“ (die Präposition), „cheval“ (das Nomen „Pferd“), und „à cheval“ (das Adjektiv „beritten“), die den kleinsten syntagmatischen Einheiten entsprechen, kann in der MAF-Notation wie in Listing 10.7 dargelegt notiert werden (entnommen aus ISO/DIS 24611, S. 28 und angepasst).

Listing 10.7: XML-Kodierung struktureller Ambiguität

```
<token id="t1">fer</token>
<token id="t2">à</token>
<token id="t3">cheval</token>
<fsm init="S0" final="S3">
 <transition source="S0" target="S3">
  <wordForm tokens="t1 t2 t3" entry="urn:lexicon:fr:fer_%E0_cheval" lemma="fer_à_cheval"/>
 </transition>
```

[4] Die Elementnamen werden in Clément und de la Clergerie (2005, S. 93) noch nicht als endgültig angesehen.

```
 <transition source="S0" target="S1">
  <wordForm entry="urn:lexicon:fr:fer" tokens="t1"/>
 </transition>
 <transition source="S1" target="S2">
  <wordForm tokens="t2" entry="urn:lexicon:fr:%E0" lemma="à"/>
 </transition>
 <transition source="S2" target="S3">
  <wordForm tokens="t3" entry="urn:lexicon:fr:cheval"/>
 </transition>
 <transition source="S1" target="S3">
  <wordForm tokens="t2 t3" entry="urn:lexicon:fr:%E0_cheval" lemma="à_cheval"/>
 </transition>
</fsm>
```

Die Attribute `init` und `final` kennzeichnen den Start- und Endknoten des FSA in Bezug auf Wortformen.[5] Unterhalb des Elements `fsm` werden mittels des `transition`-Elements die Übergänge zwischen den Zuständen (gekennzeichnet durch die Attribute `source` und `target`) kodiert, die entsprechende Wortform wird über das `wordForm`-Kindelement realisiert.

10.1.4 Metadaten

Metadaten werden in ISO/DIS 24611 nur indirekt behandelt. Unterschieden wird zwischen Metadaten, die Vorgaben zur Segmentierung im Standoff-Verfahren machen, und die über das `addressing`-Attribut des Wurzelelements `maf` gekennzeichnet werden, und allgemeinen Metadaten. Sinnvollerweise bezieht sich die Spezifikation in beiden Fällen auf vorhandene Standards wie das LINGUISTIC ANNOTATION FRAMEWORK oder OLAC (vgl. Abschnitt 12.2). Das aus ISO/DIS 24611, S. 35 entnommene Listing 10.8 zeigt die Referenz auf die Segmentierung nach MPEG7-Zeitangaben (ISO/IEC 15938-2:2002), sowie einige OLAC-Metadaten.

Listing 10.8: Metadaten zur Segmentierung

```
<?xml version="1.0" encoding="UTF-8"?>
<maf document="interview.mpeg" addressing="mpeg7">
 <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
  xmlns="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/">
  <creator>Maik Stührenberg</creator>
  <date>2011-12-14</date>
  <description>Händisch angepasstes MAF-Beispiel auf Basis des in ISO/DIS 24611, S. 35 enthaltenen
      Fragments.</description>
 </olac:olac>
 <token id="t0" from="T00:01:16:4484F30000" to="T00:01:16:14494F30000"
  transcription="mister"/>
 <wordForm tokens="t0" lemma="mister">
  <!-- ... -->
 </wordForm>
 <!-- ... -->
</maf>
```

[5] Sollen strukturelle Ambiguitäten zwischen Token notiert werden, sind `tinit` und `tfinal` zu verwenden.

10.2 Anwendung

Konkrete Anwendungen des MORPHO-SYNTACTIC ANNOTATION FRAMEWORK sind aktuell dem Verfasser nicht bekannt. Es ist aber davon auszugehen, dass die maßgeblich an der Entwicklung der Spezifikation beteiligten Akteure, hier ist vorrangig Laurent Romary zu nennen, MAF bereits zumindest testweise einsetzen. Ein Hinweis darauf findet sich auf den Webseiten des <tiger2/>-XML-Formats.[6] Als Nachfolger und Erweiterung von TIGER-XML, einem Format zur Kodierung von Baumbanken, bietet <tiger2/> Schnittstellen zur Verwendung von MAF-Sequenzierung auf Sub-Token-Ebene und erlaubt eine Serialisierung des SYNTACTIC ANNOTATION FRAMEWORK (Romary *et al.* 2011, vgl. auch Kapitel 11). Das Format wird von *INRIA* (dem französischen *Institut national de recherche en informatique et en automatique*) und der *Humboldt-Universität zu Berlin* entwickelt.

10.3 Beurteilung

Am MORPHO-SYNTACTIC ANNOTATION FRAMEWORK lässt sich gut der aktuelle Stand der internationalen Normierung ablesen: Nach den Erfahrungen mit zu spezifischen Tagsets und den daraus resultierenden alternativen Ansätzen (hier wäre beispielsweise die Abspaltung des XCES von der TEXT ENCODING INITIATIVE zu nennen, vgl. die Kapitel 5 und 7), werden neuralgische Aspekte wie die Benennung von Annotationseinheiten (seien es Elemente oder Attribute) in externe Datenkategorienregistrierungen (DCR) ausgelagert. Damit bleiben konzeptuelle Ebene und Serialisierung nachvollziehbar voneinander getrennt (vgl. Abschnitt 8.3). Die Annotation als solche erfolgt über allgemeine Mechanismen wie die standardisierte Beschreibung von Merkmalsstrukturen, was sowohl Vor- als auch Nachteile mit sich bringt. Bański und Przepiórkowski (2010a, S. 100) fassen diesen Trend wie folgt zusammen: „A tendency may be observed of increasing abstractness and generality of proposed standards [...]. This leads to their greater formal elegance, at the cost of their actual usefulness.“ Da die Spezifikation allerdings noch nicht abgeschlossen ist, kann eine endgültige Aussage hierzu nicht getroffen werden. Dieser Umstand ist insofern problematisch, da MAF und das in Kapitel 11 diskutierte SYNTACTIC ANNOTATION FRAMEWORK eng miteinander zusammenhängen. Ein Auseinanderdriften der beiden Standards wäre alles andere als wünschenswert.

Positiv hervorzuheben ist, dass ISO/DIS 24611 sinnvolle Verknüpfungen zu anderen ISO-Normen (sowohl in der Entwicklung befindliche als auch verabschiedete) herstellt, und eine mögliche konkrete Nutzung der Internationalen Standards ISO 24610-1:2006 und ISO 12620:2009 aufzeigt. Ebenso kann der Standard als Konkretisierung des allgemeineren LINGUISTIC ANNOTATION FRAMEWORK aufgefasst werden, beide Spezifikationen nutzen das formale Modell des Graphen und ähneln sich auch vom Metamodell stark. In diesem Zusammenhang wäre allerdings die Verwendung des allgemeinen Austauschformats GRAF wünschenswert gewesen. Denkbar ist aber, dass bis zur endgültigen Verabschiedung der Norm, analog zum Vorgehen beim LINGUI-

[6] Zu finden unter `http://korpling.german.hu-berlin.de/tiger2/homepage/index.html`, zuletzt abgerufen am 19.04.2012.

STIC ANNOTATION FRAMEWORK und dem eng verwandten SYNTACTIC ANNOTATION FRAMEWORK (Kapitel 11), auf die Dokumentgrammatik vollständig verzichtet wird.

Für die vorliegende aktuelle Version ISO/DIS 24611 gilt, dass die Auswahl von RELAX NG als Constraint Language nicht auf Basis der formalen Ausdrucksstärke, sondern aufgrund dessen Status als Internationaler Standard erfolgt. Die Verwendung dieser Schemasprache ist aber auch aus technischen Gründen sinnvoll, da die Dokumentgrammatik optionale Gruppen von Attributen enthält, die so direkt nicht von XML SCHEMA unterstützt werden. Aus formaler Sicht gibt es dazu keine Notwendigkeit, da das Schema einer lokalen Baumgrammatik (LTG) entspricht, was natürlich auch der Tatsache geschuldet ist, dass die Dokumentgrammatik insgesamt nicht sehr umfangreich ist. Allerdings wird in ISO/DIS 24611, S. 37 darauf hingewiesen, dass einige semantische Constraints durch das Schema nicht überprüft werden. Hier wird explizit die strukturelle Ambiguität der Token genannt (vgl. Abschnitt 10.1.3), die sich prinzipiell mit Hilfe von Co-Constraints (vgl. Abschnitt 3.4.4.4) kontrollieren lassen würde. Da die Normierung noch nicht abgeschlossen ist, bleibt abzuwarten, ob das Schema auch in der finalen Fassung der Spezifikation enthalten sein wird und in wieweit das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK in der wissenschaftlichen Gemeinschaft akzeptiert werden wird.

11 Syntactic Annotation Framework

Ähnlich wie das in Kapitel 10 diskutierte MORPHO-SYNTACTIC ANNOTATION FRAMEWORK nutzt auch das eng verwandte SYNTACTIC ANNOTATION FRAMEWORK (SYNAF) generische Annotationsmechanismen zur Auszeichnung syntaktischer Merkmale, ohne ein spezifisches Tagsets vorzugeben (Declerck 2006, S. 229). Somit ist die 2010 als internationale Norm ISO 24615:2010 verabschiedete Spezifikation (die letzte frei verfügbare Fassung ist ISO/FDIS 24615) als Ergänzung zum in Kapitel 10 diskutierten MORPHO-SYNTACTIC ANNOTATION FRAMEWORK zu sehen (Declerck 2008, S. 3025). Auf Basis bestehender Initiativen im Bereich syntaktischer und morphosyntaktischer Annotation (als Beispiele werden u. a. die Projekte „EAGLES“ und „MULTEXT“, sowie die Baumbanken-Tagsets NEGRA/TIGER und ITALIEN SYNTAX-SEMANTIC TREEBANK (ISST) genannt, vgl. Declerck *et al.* 2006) wurde das SYNTACTIC ANNOTATION FRAMEWORK als New Work Item Proposal (vgl. Abschnitt 2.1.1) unter der Leitung von Thierry Declerck eingebracht (Declerck 2006, S. 229). Ziel ist nicht nur die Repräsentation bestehender syntaktischer Annotationen (z. B. durch Transformation), sondern auch die Verwendung als zukünftiges Exportformat linguistischer Werkzeuge.

11.1 Aufbau

Der Standard definiert ein Metamodell zur syntaktischen Annotation in Verbindung mit Datenkategorien zur Verwendung in einer entsprechenden Registrierung (vgl. Kapitel 8) und wurde primär im Projekt „LIRICS“ entwickelt.[1] Die verwendeten Datenkategorien werden analog zur Vorgehensweise des MORPHO-SYNTACTIC ANNOTATION FRAMEWORK in Form eines Data Category Sets (DCS) zur Verfügung gestellt. Im Gegensatz zum MORPHO-SYNTACTIC ANNOTATION FRAMEWORK, das die Annotation von POS-, morphologischen und grammatikalischen Informationen zum Ziel hat, können mit Hilfe

[1] Der Projekttitel steht für „Linguistic Infrastructure for Interoperable Resources and Systems“. Nähere Informationen sind zu finden unter `http://lirics.loria.fr`, zuletzt abgerufen am 19.04.2012.

von SynAF Annotationen der syntaktischen Konstituenten (in Form von Gruppen morphosyntaktisch annotierter Einheiten) innerhalb der Satzgrenzen standardisiert vorgenommen werden (Declerck 2006, S. 229). Dabei sollen sowohl die linguistischen Konstituenten (wie z. B. NPs) als auch Dependenzstrukturen abgebildet werden können (ISO/FDIS 24615, S. 9). Dependenzstrukturen können zwischen morphosyntaktisch annotierten Einheiten (beispielsweise nach Vorgabe des Morpho-syntactic Annotation Framework) *innerhalb* einer Phrase (z. B. ein Adjektiv, das das Kopfnomen einer NP modifziert) oder zwischen Haupt- und Nebensatz auftreten (z. B. eine NP als „Subjekt" des Vollverbs eines Satzes), wobei die erste Ebene als *Internal Dependency*, die zweite als *External Depency* bezeichnet wird (Declerck 2006, S. 230). Zusammengefasst definiert die Norm damit ein Framework zur Mehr-Ebenen-Annotation (Declerck 2008, S. 29). Ausgehend von der Wortform (im Sinne des Morpho-syntactic Annotation Framework, vgl. Kapitel 10) werden sogenannte Syntaktische Knoten (*SyntacticNode*) etabliert, die ihrerseits in terminale und nicht-terminale Knoten (*T_Node* bzw. *NT_Node*) unterteilt werden können.[2] Die Menge der T_Nodes entspricht der Gesamtheit der terminalen Knoten in einer Baumrepräsentation der syntaktischen Struktur, d. h. den morphosyntaktisch annotierten Wortformen und evtl. leeren Elementen, jeweils definiert durch Segmente in den Primärdaten (analog zu den Token in MAF, vgl. Kapitel 10). Jeder T_Node enthält die syntaktische Annotation der durch ihn repräsentierten Konstituenten. Die nicht-terminalen Knoten im Syntaxbaum (NT_Node) repräsentieren syntaktische Einheiten auf Phrasen-Ebene (oder höher, einschließlich des gesamten Satzes) und sind mit den entsprechenden syntaktischen Kategorien annotiert.

Die Menge der SyntacticNodes bildet zusammen mit den Kanten (*SyntacticEdge*) einen Syntax-Graphen (*SyntacticGraph*), wobei die Kanten Relationen zwischen den Knoten abbilden. Diese Beziehungen können verschiedener Art sein (beispielsweise die binäre Dependenzrelation zwischen einem Quell- und einem Zielknoten) und wie die Knoten über Annotationen typisiert werden. Problematisch in diesem Zusammenhang ist, dass in ISO/FDIS 24615 jegliche Angaben welcher Art diese Annotationen sein sollen fehlen.[3]

Abbildung 11.1 zeigt eine vereinfachte und übersetzte Version des in ISO/FDIS 24615 enthaltenen Metamodells. Die gestrichelten Pfeile zeigen eine optionale Beziehung an (so ist ein Syntax-Graph auch ohne Kanten gültig und die Annotation der Kanten ist ebenfalls nicht obligatorisch). Zu beachten ist, dass die Wortformen außerhalb des Skopus der Norm liegen, da sie Gegenstand des in Kapitel 10 diskutierten Morpho-syntactic Annotation Framework sind. In der Grafik nicht aufgeführt sind die weiteren MAF-Komponenten (vgl. auch Abbildung 10.1 auf Seite 216).

Die Typisierung erfolgt ebenso wie die Annotation der syntaktischen Merkmale über die von der Spezifikation vorgegebene DCS (ISO/FDIS 24615, S. 6). Auch hier wurden Anknüpfungen an eine Reihe vorhandener Annotationsschemata hergestellt, um grundlegende Kategorien zu identifizieren (Declerck 2008, S. 29). Die Datenkategorien selbst befinden sich im normativen Anhang A der Spezifikation, wobei diese nach Bedarf durch

[2] Da der Standard keine Serialisierung bzw. ein konkretes Exportformat vorschreibt, handelt es sich nicht um Elementnamen, sondern prinzipiell um Klassen von Knoten in einer Graphenrepräsentation.

[3] Es wird zwar im Abschnitt „Normative References" auf ISO 24610-1:2006 verwiesen, im sonstigen Text gibt es aber keinerlei erläuternde Erklärung hinsichtlich der Strukturierung der Annotation bzw. der Art der Beziehung zu ISO 24610-1:2006.

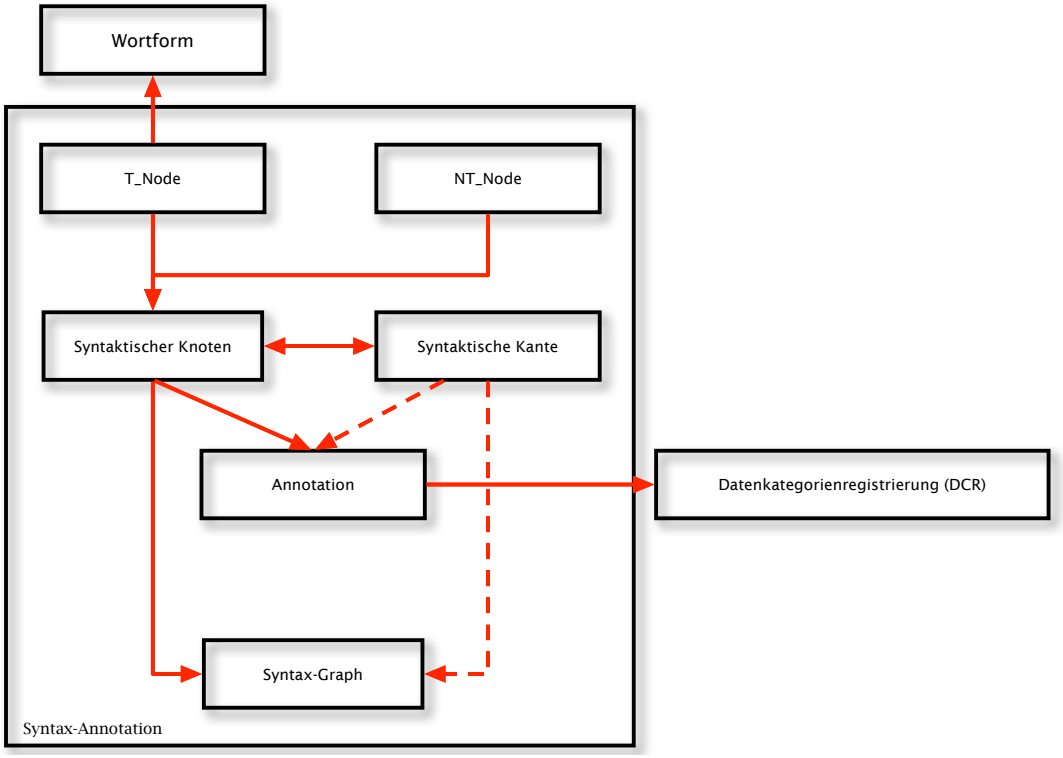


Abbildung 11.1: Syntactic Annotation Framework-Metamodell

zusätzliche Kategorien erweitert werden können. Unterschieden werden drei Klassen von Datenkategorien: die grundlegenden syntaktischen Kategorien (*Basic Syntactic Data Categories*), Kategorien zur Darstellung der Konstituentenstruktur (*Constituency Related Data Categories*) und solche zur Annotation der Dependenzstruktur (*Dependency Related Data Categories*). Zur ersten Gruppe gehören insgesamt 33 Kategorien, neben eher allgemeinen Angaben wie Inline- oder Standoff-Annotation (*/embeddedNotation/* bzw. */standoffNotation/*) oder */parsing/* auch die Kategorien */primarySyntacticEdge/* und */secondarySyntacticEdge/*, die mögliche Werte der Kategorie */syntacticEdgeType/* sind, die zur Typisierung von Kanten im Syntax-Graph herangezogen wird.[4] Als primäre syntaktische Kante wird die direkte Kante zwischen zwei Knoten in der Konstituentenstruktur zwischen Konstituente und deren Komponente bezeichnet. Dagegen ist eine sekundäre Kante im Syntax-Graphen nur indirekt zwischen zwei Konstituenten vorhanden. Als Beispiel wird in ISO/FDIS 24615, S. 9 die Beziehung zwischen dem Kopf einer Phrase und einem Koreferenten genannt.

Die Kategorien zur Darstellung der Konstituentenstruktur umfassen 28 Einträge. Hier findet sich */grammaticalUnit/* mit den Unterkategorien */chunk/*, */clause/*, */phrase/* und */sentence/* zur Typisierung der Konstituente. Während sich grammatikalische Einheiten zur Kategorie */syntacticConstituent/* zusammenfassen lassen, können sie auch weiter spezifiziert werden, beispielsweise als */prepositionPhrase/*, */relativeClause/*

[4] Zusätzliche Werte sind ebenfalls möglich, aber nicht im der Spezifikation beschrieben.

oder */nounChunk/*.

Die Angaben zur Dependenzstruktur sind in 33 Kategorien unterteilt. Oberkategorien sind hier */relation/* (eine annotierte Beziehung zwischen mindestens zwei grammatikalischen Einheiten, weiter typisierbar beispielsweise als */comparativeRelation/* oder */relativeRelation/*) und */modifier/*, letztere lässt sich näher spezifizieren durch Kategorien wie */adverbModifier/*, */verbModifier/* oder */postnominalModifier/*.

Die vollständige DCS findet sich bereits in der DCR-Implementierung ISOcat (vgl. Abschnitt 8.2) als öffentlich zugängliche Kategorienauswahl „Syntax“ (*Thematic Domain Group 4*, TDG 4), so dass an dieser Stelle nicht näher darauf eingegangen wird.[5] Abschließend bleibt festzustellen, dass die Auswahl an Datenkategorien recht erschöpfend wirkt. Sollten dennoch im Einzelfall Merkmale fehlen oder unterspezifiziert sein, bleiben die in Kapitel 8 beschriebenen Möglichkeiten der Erweiterung.

11.2 Serialisierung

Da die Spezifikation nur das Metamodell sowie eine DCS definiert, ist ein normatives (oder wenigstens informatives) offizielles Annotationsformat für das Syntactic Annotation Framework nicht vorhanden. Die beiden in Declerck (2006) aufgeführten Annotationsschemata Tiger und ISST bilden nur Teile des SynAF-Metamodells ab und dienen daher nur als Ausgangslage für die weitere Entwicklung. Wie allerdings bereits in Abschnitt 10.2 angedeutet und in Romary *et al.* (2011) beschrieben, kann <tiger2/> (eine Weiterentwicklung des Tiger-Formats) als ein mögliches Serialisierungsformat für SynAF-modellierte Annotation angesehen werden (ähnlich wie für MAF).

Im informativen Anhang B von ISO/FDIS 24615 wird für eine Implementierung auf das Linguistic Annotation Framework verwiesen. Allerdings sind diese Ausführungen nur sehr kurz abgefasst und stützen sich vorrangig auf das Pivot Format, also GrAF. Die in diesem Abschnitt enthaltene Abbildung B.1, die auch in ISO/DIS 24612 oder in Ide und Suderman 2007 enthalten ist, und die Verwendung des Pivot Formats verdeutlicht, ist nicht mehr Bestandteil der aktuellen Version der LAF-Spezifikation (ISO/FDIS 24612). Dazu kommt, dass für GrAF in diesem FDIS nur ein Datenmodell, nicht aber eine konkrete Serialisierung vorgegeben ist.

Hayashi *et al.* (2010) präsentieren ein auf GrAF basierendes Annotationsformat zur Serialisierung von SynAF, das die aus Ide und Suderman (2007) bekannte Fassung von GrAF um die Darstellung von Dependenzstrukturen erweitert, ähnlich wie das von Kountz *et al.* (2008) propagierte Format (vgl. Abschnitt 9.3). Allerdings ist auch diese Version nicht mehr aktuell. Unklar bleibt, warum in ISO/FDIS 24615 im Abschnitt „Normative references“ ISO 24610-1:2006 aufgeführt wird, ein Verweis auf das Linguistic Annotation Framework hier aber fehlt. Es kann davon ausgegangen werden, dass die Annotation der syntaktischen Informationen in Form der standardisierten Mermalsstrukturbeschreibung erfolgt.

[5] Im Gegensatz zu den in ISO/FDIS 24615 aufgeführten 94 Kategorien sind in ISOcat 102 enthalten. Zu den zusätzlich vorhandenen zählen u. a. */valency/*, */verbFrame/* oder */interrogative/*.

11.3 Beurteilung

Die Beurteilung des SYNTACTIC ANNOTATION FRAMEWORK fällt nicht leicht. Sowohl das Datenmodell als auch die DCS sind gut strukturiert und flexibel einsetzbar. Der Standard als solcher beschränkt sich damit auf das Nötigste und ist dadurch von einer Reihe anderer Normen abhängig. Problematisch ist in diesem Zusammenhang das Fehlen einer Standard-konformen Serialisierung, da das LINGUISTIC ANNOTATION FRAMEWORK weiterhin nicht als Internationaler Standard verabschiedet ist. Ähnlich sieht die Situation bezüglich der Relation zu den Normen ISO 24610-1:2006 und ISO/DIS 24611 aus. Auch hier bezieht sich SYNAF in weiten Teilen (inkl. der Segmentierung in Form von Wortformen) auf eine zur Zeit nicht in finaler Form verabschiedete Norm. Allerdings zeigen Beispiele des <tiger2/>-Formats, dass zumindest Teile des SYNTACTIC ANNOTATION FRAMEWORK bereits jetzt schon einsetzbar sind. Es bleibt abzuwarten, inwieweit die endgültigen Fassungen der referenzierten Normen die Kompatibilität zu SYNAF berücksichtigen.

Metadata is machine understandable information about web resources or other things.

Berners-Lee (1997)

12 Metadaten – Daten über Daten

Bei Metadaten handelt es sich um maschinenlesbare (strukturierte) Daten, die dazu dienen, eine Informationsressource zu beschreiben oder, um es einfacher auszudrücken: um „Daten über Daten“ (Lemnitzer und Zinsmeister 2006, S. 45). Darunter fallen grundlegende Informationen wie z. B. Titel, Autor und Angaben über Ort und Zeit der Veröffentlichung, aber auch speziellere Informationen, die auf einen kostengünstigen und effektiven Einsatz in digitalen Netzen abzielen und es damit zum Teil erst ermöglichen, sinnvolle Navigations- und Retrievalstrategien (wie z. B. Suchindex oder Inhaltsverzeichnis) anzubieten. So führt Burnard (2005, S. 30) aus: „[metadata is] the kind of data that is needed to describe a digital resource in sufficient detail and with sufficient accuracy for some agent to determine whether or not that digital resource is of relevance to a particular enquiry.“

Die Gesamtheit der Metadaten lässt sich unterteilen in administrative, redaktionelle, analytische und deskriptive Informationen (Burnard 2005), was beispielsweise in den TEI Guidelines (vgl. Kapitel 5) durch die Strukturierung der Elemente `fileDesc` und `encodingDesc` verdeutlicht wird.

Good (2002) folgend, ist das Konzept von Metadaten nicht wirklich neu, allerdings ist die Art und Weise, wie Informationen über ein Objekt (oder eben: Daten über Daten) gespeichert werden, entscheidend. Es kann unterschieden werden zwischen Metadaten, die für eine Sammlung physischer Objekte (wie z. B. gedruckte Werke) erforderlich sind, und Metadaten, die zur Verwaltung digitaler Objekte erforderlich sind. Das gilt gerade dann, wenn digitale Abbildungen physischer Objekte zum Einsatz kommen:

> Wenn eine Bibliothek Metadaten zu einem Buch in ihrem Bestand erfasst, wird dieses Buch nicht in eine Reihe einzelner Blätter zerfallen, weil keine Strukturangaben über die innere Ordnung des Buches erhoben werden. Noch werden Forscher das Buch schlechter nutzen können, wenn nicht angegeben wurde, dass es mit einer Ryobi Druckmaschine hergestellt wurde. Gleiches gilt jedoch nicht für die digitale Version desselben Buches. Ohne Metadaten

> zur Struktur sind die Seitenabbildungen oder die Textdateien, aus denen es besteht, so gut wie wertlos. (The Library of Congress 2005).

Dadurch, dass die Kapazitäten heutiger Rechnersysteme immer größere linguistische Korpora erlauben (sowohl was die Größe der Daten als auch die Dichte ihrer Annotation angeht), kommt der Verwendung von Metadaten eine immer stärkere Bedeutung zu, denn nur sie erlauben es, innerhalb einer Menge von Daten die entsprechenden Informationen zu finden, die für eine bestimmte Aufgabenstellung benötigt werden. Darüber hinaus ermöglichen sie die Wahrnehmung der eigentlichen Daten in einem größeren Umfeld (Good 2002).

Einige der bisher in diesem Teil diskutierten Spezifikationen machen eigene Vorgaben in Bezug auf die Strukturierung und Informationen der Metadaten (beispielsweise TEI, vgl. Kapitel 5 und XCES, vgl. Kapitel 7). Andere wiederum verweisen auf eigenständige Metadaten-Standards, von denen in diesem Kapitel einige in Kurzform vorgestellt werden. Dabei ist zu unterscheiden zwischen generischen Metadaten (z. B. DUBLIN CORE) und solchen, die für bestimmte Disziplinen entwickelt wurden. Die zweite Gruppe ist oftmals eine Übermenge oder aber Konkretisierung der ersten (als Beispiel sei der Standard LEARNING OBJECT METADATA, IEEE 1484.12.1-2002, genannt). Im Gegensatz zu den anderen in diesem Teil der Arbeit detailliert vorgestellten und bewerteten Spezifikationen erfolgt in diesem Kapitel keine genauere Diskussion der Standards. Vielmehr soll eine Kurzübersicht über relevante Metadaten-Spezifikationen und damit ein erster Orientierungsrahmen gegeben werden. Dabei wird kein Anspruch auf Vollständigkeit erhoben.

12.1 Generische Metadaten: Dublin Core

DUBLIN CORE (DC, genauer: das DUBLIN CORE METADATA ELEMENT SET, DCMES) kann als kleinster gemeinsamer Nenner im Bereich der Metadaten angesehen werden. Die Spezifikation, die sowohl als amerikanische Norm ANSI/NISO Z39.85-2007 als auch als RFC 5013 (*Request for Comments*)[1] und Internationaler Standard ISO 15836:2009 publiziert wurde, ist seit über zehn Jahren in der Version 1.1 verfügbar (mit kleineren Änderungen, die sich alle über die Webseite der *Dublin Core Metadata Initiative*, *DCMI*, einsehen lassen) und definiert ein Vokabular von 15 Datenkategorien. Damit gilt DC als Vorgänger von Datenkategorisierungen wie ISO 12620:2009 (vgl. Kapitel 8). Aus dem Namensbestandteil „Core“ wird deutlich, dass es sich hierbei um allgemeine, grundlegende Konzepte handelt, die für nahezu alle Arten von Daten verwendet werden können.[2] Das und die Tatsache, dass alle DC-Spezifikationen vollständig mittels RDF implementiert sind (Nilsson *et al.* 2008), macht die Spezifikation sehr einfach

[1] RFCs wurden ursprünglich als Diskussionsgrundlage ins Leben gerufen, haben sich aber in einigen Bereichen des Internets durch ihre Nutzung als Quasi-Standards etabliert (z. B. das HYPERTEXT TRANSFER PROTOCOL, HTTP, das in RFC 1945 und RFC 2616 definiert ist). Weitere Informationen sind zu finden unter `http://www.rfc-editor.org/`, zuletzt abgerufen am 19.04.2012.

[2] Daher wird das DCMES auch als „Simple Dublin Core“ bezeichnet (Beckett *et al.* 2002) im Gegensatz zu „Qualified Dublin Core“.

nutzbar.[3] So gibt es Möglichkeiten, DUBLIN CORE-Metadaten in (X)HTML-Instanzen einzubetten (Johnston und A. Powell 2008) und auch nahezu alle anderen Metadaten-Standards sind insofern kompatibel zu DUBLIN CORE, als dass sie Obermengen oder Erweiterungen darstellen. Der Einsatz solcher Verfeinerungen (*Refinements*, d. h., die Kennzeichnung einer Kategorie in Abhängigkeit einer vorhandenen anderen) oder die nähere Spezifikation durch Vokabulare oder andere Kodierschemata ist ebenfalls im Standard spezifiziert. Die 15 Datenkategorien (intern „Element“ oder „Term“ genannt) sind *Contributor*, *Coverage*, *Creator*, *Date*, *Description*, *Format*, *Identifier*, *Language*, *Publisher*, *Relation*, *Rights*, *Source*, *Subject*, *Title* und *Type*. Alle Elemente des DCMES werden – wie bei ISO 12620:2009 – über einen URI angesprochen, die sich zusammensetzt aus dem Namensraum `http://purl.org/dc/elements/1.1/` und dem Termnamen in Kleinbuchstaben, also `http://purl.org/dc/elements/1.1/contributor` für die Kategorie *Contributor*.

Die aktuelleren DCMI METADATA TERMS ergänzen die genannten Kategorien um weitere 40, darunter rechtliche (z. B. *accessRights*, *dateCopyrighted*, *license*) oder didaktische Angaben (*instructionalMethod*, *educationLevel*).[4] Dabei spezifizieren die zusätzlichen Kategorien die ursprünglichen näher, beispielsweise ist die neue Kategorie *Abstract* ein Refinement der Kategorie *Description*, *Alternative* von *Title*. Zusätzlich verweisen die DCMI METADATA TERMS auf weitere Spezifikationen, die zur näheren Bestimmung der Kategorien bzw. zur Eintragung von Werten herangezogen werden können – beispielsweise ISO 639-3:2007 als Datentypen zur Bestimmung von Sprachen – und definiert Klassen zur Gruppierung von Eigenschaften. Aus den genanten Gründen sind vor allem die DCMI METADATA TERMS gut unterstützte und nachhaltig verwendbare generische Metadaten-Standards, die als Grundlage für linguistische Korpora dienen können.

12.2 The Open Language Archives Community Metadata

Ausgehend von den Arbeiten der *DCMI* wurde im Oktober 1999 die *Open Archives Initiative* (*OAI*) mit dem Ziel gegründet, ein grundlegendes Framework zu entwickeln, das digitale Inhalte verschiedener Art über verteilte Repositories (in Form von Servern) zugänglich machen soll.[5] Zu diesem Zweck wurde neben einem auf DUBLIN CORE aufbauenden Satz an Metadaten-Elementen auch das OAI PROTOCOL FOR METADATA HARVESTING (OAI-PMH), ein auf HTTP basierendes Protokoll zur Abfrage von Software-Diensten, entwickelt. Ausgehend von diesen Arbeiten entwickelte die im Dezember 2000 gegründete *Open Language Archives Community*, ein internationaler Zusammenschluss

[3] Zusätzliche mittels XML SCHEMA modellierte Dokumentgrammatiken sind ebenfalls vorhanden und unter `http://dublincore.org/schemas/xmls/` einsehbar. Zuletzt abgerufen am 19.04.2012.

[4] Um genau zu sein, sind die 15 bereits bekannten Kategorien nicht dieselben wie in DCMES, da sie einem anderem Namensraum (`http://purl.org/dc/terms/` vs. `http://purl.org/dc/elements/1.1/`) zugeordnet werden und teilweise Domänen und Bereiche besitzen. So stellt `http://purl.org/dc/terms/contributor` eine Verfeinerung von `http://purl.org/dc/elements/1.1/contributor` dar, indem es dem Bereich *Agent* (`http://purl.org/dc/terms/Agent`) zugeordnet wird. Ein Agent ist dabei ein Aktor, also eine Ressource (Mensch oder Maschine), die eine Handlung vornehmen kann.

[5] Nähere Informationen sowohl zur Organisationsstruktur als auch den Zielen und Spezifikationen finden sich unter `http://www.openarchives.org/`, zuletzt abgerufen am 19.04.2012.

von Institutionen und Individuen aus dem sprachwissenschaftlichem Bereich, Erweiterungen in Form zusätzlicher Metadatenkategorien unter Beibehaltung des OAI-PMH, um Sprachressourcen zugänglich zu machen (Bird und Simons 2001; Simons und Bird 2003). Das Resultat dieser Bemühungen ist die schlicht als OLAC METADATA titulierte Spezifikation (Simons und Bird 2008a).[6] Als *OLAC*-interner Standard wird er von allen Archiven und Dienstleistern der *Open Language Archives Community* angewendet. Neben allen 15 Datenkategorien aus DCMES (Bird und Simons 2003a) werden mittels Refinement Metadaten weiter qualifiziert. OLAC METADATA nutzt XML SCHEMA zur Definition eines gradlinigen Metadatenformats, Listing 12.1 zeigt eine Beispiel-Instanz.[7] Die Zeile 17 zeigt ein Beispiel für eine nähere Spezifizierung mittels Refinement, indem über den Wert *olac:linguistic-type* des `xsi:type`-Attributs die Ressource zunächst grundlegend als linguistisch kategorisiert wird und diese Kategorie durch das Attribut `olac:code` bzw. dessen Wert *language_description* als Sprachbeschreibung (hier: einer Kayardild-Grammatik) ausgezeichnet wird.[8] In Zeile 18 dagegen ist ein Beispiel für eine Erweiterung der DCMES-Datenkategorie *Type* durch die DCMI METADATA TERMS zu finden.

Listing 12.1: OLAC-Instanz

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.1/"
3   xmlns="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://www.language-archives.org/OLAC/1.1/ http://www.language-archives
        .org/OLAC/1.1/olac.xsd">
6   <title>A grammar of Kayardild. With comparative notes on Tangkic.</title>
7   <creator>Evans, Nicholas D.</creator>
8   <subject>Kayardild grammar</subject>
9   <subject xsi:type="olac:language" olac:code="gyd">Kayardild</subject>
10  <language xsi:type="olac:language" olac:code="en">English</language>
11  <description>Kayardild Grammar (ISBN 3110127954)</description>
12  <publisher>Berlin – Mouton de Gruyter</publisher>
13  <contributor xsi:type="olac:role" olac:code="author">Nicholas Evans</contributor>
14  <format>hardcover, 837 pages</format>
15  <relation>related to ISBN 0646119966</relation>
16  <coverage>Australia</coverage>
17  <type xsi:type="olac:linguistic-type" olac:code="language_description"/>
18  <type xsi:type="dcterms:DCMIType">Text</type>
19 </olac:olac>
```

Die Verfeinerungen gegenüber DCMES sind gruppiert nach Rollen (definiert im XML-Schema `olac-role.xsd` und erläutert in Johnson 2006) und Art der Ressource (defi-

[6] Daneben existieren mit OLAC REPOSITORIES (Simons und Bird 2008b) und OLAC PROCESS (Simons und Bird 2006) weitere *OLAC*-Standards, die den Aufbau eines *OLAC*-Archivs sowie der Organisation definieren. Weiterhin geben Empfehlungen (*Recommendations*) grundlegende Hinweise, die aber nicht zwingend beachtet werden müssen.

[7] Insgesamt handelt es sich um sieben XSD-Instanzen, zuzüglich der drei importierten XML-Schemata für DC. Die Hauptdatei `olac.xsd` ist zu finden unter `http://www.language-archives.org/OLAC/1.1/olac.xsd`. Die in Listing 12.1 gezeigte Instanz ist entnommen aus Simons und Bird (2008a) und ist einsehbar unter der URL `http://www.language-archives.org/OLAC/1.1/olac.xml`. Beide URLs zuletzt abgerufen am 19.04.2012.

[8] Neben dem Wert *language_description* sieht das XML-Schema `olac-linguistic-type.xsd` (abrufbar unter `http://www.language-archives.org/OLAC/1.1/olac-linguistic-type.xsd`, zuletzt abgerufen am 19.04.2012) die Werte *lexicon* und *primary_text* vor.

niert in den XML-Schemata `olac-discourse-type` und `olac-linguistic-field.xsd` und erläutert in Johnson und Dry 2006; Dry 2006). Dazu kommen nähere Angaben in Bezug auf die Sprache (definiert im XML-Schema `olac-language.xsd`) und OLAC-eigene und fremde Erweiterungen (definiert im XML-Schema `olac-extension.xsd`, vgl. auch Bird und Simons 2003b). Erweiterungen werden generell über das `xsi:type`-Attribut in der Instanz markiert, so dass das OLAC-Schema nicht angepasst werden muss. Vielmehr verweist die Instanz über das `xsi:schemaLocation`-Attribut auf die XSD-Datei, die die Erweiterungen beinhaltet (und die ihrerseits das OLAC-Schema importiert). Ein Beispiel dafür ist in Simons und Bird (2008a, Abschnitt 5, „Defining a third-party extension“) aufgeführt. Die Angabe der XML NAMESPACES variiiert je nach Speicherung der Metadatensätze: Werden diese in einem Repository gesichert (Simons und Bird 2008b), so kann die Angabe des Standardnamensraums `http://www.language-archives.org/OLAC/1.1/` entfallen (Simons und Bird 2008a).

Durch die Anpassung der DCMI METADATA TERMS speziell an linguistische Erfordernisse ist OLAC eine sehr gut geeignete Wahl zur Strukturierung allgemeiner Metadaten für linguistische Korpora und wird zu diesem Zweck auch in ISO/DIS 24611, S. 11 empfohlen.

12.3 ISLE Metadata Initiative (IMDI)

Im Rahmen der Konferenz „International Language Resources and Evaluation“ (LREC) fand im Jahr 2000 der „First EAGLES/ISLE Workshop on Meta-Descriptions and Annotation Schemes for Multimodal/Multimedia Language Resources and Data Architectures and Software Support for Large Corpora“ unter Einbeziehung der Projekte „Expert Advisory Group on Language Engineering Standards“ (EAGLES) und „International Standard in Language Engineering“ (ISLE) statt (vgl. auch Wittenburg *et al.* 2000).[9] Ausgehend von den Ergebnissen dieses Treffens fanden zwei Jahre später auf der LREC-Konferenz 2002 Vorträge zweier Gruppen statt: einer der *Open Language Archives Community* (*OLAC*, vgl. Abschnitt 12.2) und einer der *ISLE Metadata Initiative* (IMDI, Broeder, Declerck *et al.* 2004, S. 369). Die von letzterer entwickelten Spezifikationen METADATA ELEMENTS FOR SESSION DESCRIPTIONS (IMDI Part 1) und METADATA ELEMENTS FOR CATALOGUE DESCRIPTIONS (IMDI Part 1 B, kurz: IMDI) basieren im Gegensatz zu OLAC nicht auf DUBLIN CORE, sondern orientieren sich an den Bedürfnissen von Feldforschern und anderen Sprachwissenschaftlern in Bezug auf die Katalogisierung multimodaler Daten:

> The focus was primarily on multimedia/multimodal corpora and a more detailed set was worked out that can be used not only for resource discovery but also for exploitation and managing large corpora. Most importantly, IMDI allows its metadata descriptions to be organized into linked hierarchies

[9] Weitere Angaben zum Workshop finden sich unter `http://www.mpi.nl/ISLE/events/LREC%202000/LREC2000.htm`. Die Ergebnisse des Workshops sind in einer zweiseitigen Zusammenfassung aufgeführt, die unter `http://www.mpi.nl/ISLE/documents/papers/LREC2000Workshop.pdf` abrufbar ist. Beide URLs zuletzt abgerufen am 19.04.2012.

> supporting browsing and enabling data managers to carry out a variety of management tasks. (Broeder, Declerck *et al.* 2004, S. 369).

Dabei unterscheidet IMDI zwischen Katalog- und Sitzungs-Metadaten, die in jeweils eigenständigen Standards definiert sind. Erstere dienen der Katalogisierung von Sprachressourcen, d. h., dem Auffinden vorhandener multimodaler Ressourcen (wobei seit der Version 3.0 auch Textkorpora prinzipiell mit IMDI beschreibbar sind). Sitzungs-Metadaten beschreiben ein Primärdatum (eine Aufzeichnung, einen Text) und dessen Annotation, dabei sind die Datenkategorien gruppiert nach *Session*, *Project*, *Content*, *Actors*, *Resources* und *References*, wobei jede Gruppe eine ganze Reihe von Merkmalen (und Unterschemata) umfasst. Sowohl die METADATA ELEMENTS FOR CATALOGUE DESCRIPTIONS als auch die METADATA ELEMENTS FOR SESSION DESCRIPTIONS sind in Form von XML SCHEMA-Instanzen formalisiert, allerdings können auf mehreren Ebenen Metadaten über Schlüssel-Wert-Paare bzw. Attribut-Wert-Paare gespeichert werden, so dass die Datenkategorien nicht an ein festes Tagset gebunden sind und relativ frei festgelegt werden können.[10] Die vorgegeben Datenkategorien wurden dagegen gemäß ISO 12620:2009 registriert und sind Bestandteil von ISOCAT (Broeder, Declerck *et al.* 2004, S. 372). Darüber hinaus existieren Abbildungen zwischen OLAC und IMDI und ein in IMDI kodiertes Metadaten-Repository kann auch von OLAC aus referenziert werden, d. h., als *Data Provider* dienen. Neben den Schemata gibt es eine Reihe von Software-Tools (u. a. Metadaten-Editor und -Browser), die ebenfalls frei verfügbar und quelloffen sind.[11] Abbildung 12.1 auf der nächsten Seite zeigt exemplarisch den IMDI-BROWSER zur Darstellung der hierarchisch organisierten Metadaten.

Zusammenfassend kann IMDI als ähnlich bedeutsam wie OLAC (Schmidt *et al.* 2006, S. 12) und als erste Wahl in Bezug auf multi-modale linguistische Daten angesehen werden. Dafür sprechen auch die Fortführung der Arbeiten im „CLARIN"-Projekt, die Ausführungen in Schonefeld und Milde (2004) oder in Popescu-Belis und Estrella (2007), und die große Anzahl von Archiven, bei denen die Spezifikationen eingesetzt werden.[12]

[10] Die XSD-Dateien sind auf `http://www.mpi.nl/IMDI/schemas/schemas.html` zum freien Download verfügbar, zuletzt abgerufen am 19.04.2012.

[11] Eine Übersicht findet sich in Broeder und Wittenburg (2006) und unter `http://www.lat-mpi.eu/tools/imdi`, zuletzt abgerufen am 19.04.2012.

[12] Eine Übersicht hierzu befindet sich auf der Webseite `http://www.mpi.nl/IMDI/other/other.html`, zuletzt abgerufen am 19.04.2012.

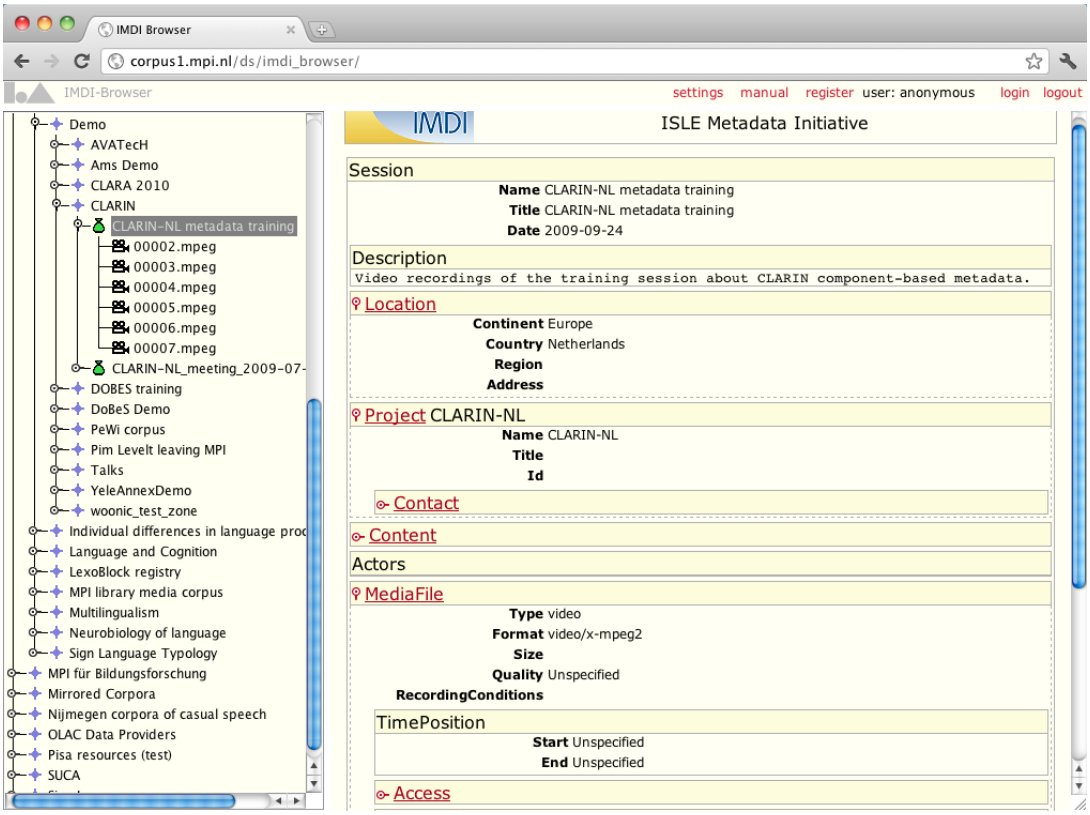


Abbildung 12.1: IMDI-Browser (Screenshot)

12.4 Weitere Arbeiten und Beurteilung

Neben den genannten Ansätzen gibt es natürlich weitere linguistisch motivierte Metadaten-Standards. Deren Skopus liegt allerdings nicht primär auf der Beschreibung von Textkorpora oder sie werden vom Verfasser aufgrund ihrer geringeren Verbreitung als nicht so relevant erachtet. Die in diesem Kapitel kurz aufgeführten Standards sind – je nach Anwendungszweck – sehr gut geeignet für die nachhaltige Katalogisierung und den Austausch linguistischer Ressourcen. Positiv anzumerken ist, dass Abbildungen der Datenkategorien zueinander aber auch zu den eingebetteten Metadaten der TEI GUIDELINES (vgl. Kapitel 5) bzw. des CORPUS ENCODING STANDARD (vgl. Kapitel 7) oftmals einfach zu bewerkstelligen sind.[13] Weitere Planungen für eine vereinheitlichte Metadatenspeicherung sind in Cieri *et al.* (2010) skizziert. Die Kombination eines der in diesem Teil dieser Arbeit diskutierten Modelle und Formate zur Strukturierung der inhaltlichen Annotation mit einem der in diesem Kapitel vorgestellten Metadaten-Standards (unter Einbezug einer Datenkategorienregistrierung wie ISOCAT) dürfte mittelfristig die sinnvollste Grundlage zur nachhaltigen Speicherung linguistischer Daten sein. In

[13] Ein erster – wenn auch schon älterer – Ansatzpunkt für einen Vergleich etablierter Metadaten-Spezifikationen bietet `http://www.mpi.nl/IMDI/overview/Overview_Summary.html`. Zuletzt abgerufen am 19.04.2012.

diesem Zusammenhang sei erneut auf das bereits in Abschnitt 8.2 genannte Framework COMPONENT METADATA INFRASTRUCTURE (CMDI) verwiesen, das eine größere Freiheit der Struktur der Metadaten erlaubt, aber auch die Verwendung bestehender Metadatenkategorien und eine Anbindung an eine DCR wie ISOCAT beinhaltet. Zumindest für rein textuelle Daten sind darüber hinaus die in den TEI GUIDELINES enthaltenen Metadaten oftmals ausreichend und als adäquater Ersatz anzusehen.

Teil III

XStandoff als Hybrid-Ansatz zur Speicherung mehrfach annotierter linguistischer Korpora

13
SGF und XStandoff

In dieser Arbeit wurde bereits eine Reihe an Spezifikationen und Standards vorgestellt, die zur strukturierten Annotation linguistischer Daten herangezogen werden können. In diesem Teil soll eine Meta-Auszeichnungssprache vorgestellt werden, die vom Verfasser entwickelt wurde und in Kombination mit anderen bisher diskutierten Ansätzen genutzt werden kann: XStandoff.

XStandoff ist kein Tagset, also kein Vokabular von Elementen der Informationsstrukturierung, sondern – wie bereits angedeutet – ein Metaformat zur strukturierten Speicherung multipler Annotationen, ähnlich dem in Kapitel 9 diskutierten Linguistic Annotation Framework. Der Vorläufer von XStandoff, das Sekimo Generic Format (SGF), wurde im Rahmen des „Sekimo"-Projekts als Teil der von der Deutschen Forschungsgemeinschaft (DFG) geförderten verteilten Forschergruppe 437 „Texttechnologische Informationsmodellierung" entwickelt (Stührenberg 2008). Beide Formate kombinieren Aspekte des Annotation Graph (vgl. Abschnitt 3.3.3.2) mit der Standoff-Notation (vgl. Abschnitt 3.3.2.3) und ermöglichen damit eine nachhaltige Nutzung linguistischer Daten:

> It has to be pointed out that stand-off architecture is one of the preconditions for sustainability and interoperability. A stand-off annotated LR preserves the source text in a minimally marked-up form and hence as capable of being easily extracted or processed by future versions of the current tools or by new tools. Such a resource is also easily expandable, which also adds to its attractiveness [...]. (Bański und Przepiórkowski 2010b, S. 35).

Im weiteren Verlauf dieses Kapitel wird XStandoff anhand eines Beispiels näher vorgestellt.

13.1 Ein Beispiel

Dieser Abschnitt führt ein einfaches Beispiel ein, um den aktuellen Entwicklungsstand von XSTANDOFF vorzustellen. Wie bereits angemerkt, ist XSTANDOFF aus dem SEKIMO GENERIC FORMAT entstanden, das vom Verfasser während der Laufzeit der zweiten Projektphase des „Sekimo“-Projekts entwickelt wurde. Aus der ersten Projektphase, an der der Verfasser nicht beteiligt war, erbt das Format die Adressierung von Annotationsgrenzen mittels Zeichenpositionen, die in Witt (2002a); Witt (2002b); Witt (2004) mit Hilfe der Programmiersprache Prolog umgesetzt wurde. Aus Gründen der Einfachheit wird zur Demonstration der folgende einfache Satz verwendet, der auf verschiedenen linguistischen Ebenen annotiert wird:

> The sun shines brighter.

Das Listing 13.1 zeigt die Adressen (Positionen) der einzelnen Zeichen im Text, die im späteren Verlauf zur Segmentierung genutzt werden (vgl. auch Abschnitt 10.1.1).

Listing 13.1: Nutzung der Zeichenpositionen

```
  T h e    s u n    s h i n e s    b r i g h t e r .
00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24
```

Der Buchstabe „T“ beginnt bei der Position 0 und endet an der Position 1, der Buchstabe „h“ reicht von Position 1 bis 2 usw. Dadurch, dass jedes Zeichen genau eine Position lang ist, können auch leere Elemente in der späteren Standoff-Annotation eindeutig positioniert werden (hier ist die Start- und Endposition identisch). Die in Witt (2004) vorgestellte Prolog-Faktenbasis nutzt dazu die in Listing 13.2 gezeigten Prädikate. Jedes einzelne Zeichen des zu annotierenden Textes wird in einem Prolog-Prädikat gespeichert, das die Start- und Endposition, sowie das Zeichen selbst beinhaltet.

Listing 13.2: Attribute in der Prolog-Faktenbasis

```
1 pcdata_node(0, 1, 'T').
2 pcdata_node(1, 2, 'h').
```

Der Beispielsatz wird nun auf zwei Annotationsebenen inline annotiert, auf der Ebene der Morpheme und auf der Silbenebene (Listing 13.3 und 13.4 auf der nächsten Seite).

Listing 13.3: Morphem-Annotation

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <morphemes>
3  <m>The</m>
4  <m>sun</m>
5  <m>shine</m>
6  <m>s</m>
7  <m>bright</m>
8  <m>er</m>.
9 </morphemes>
```

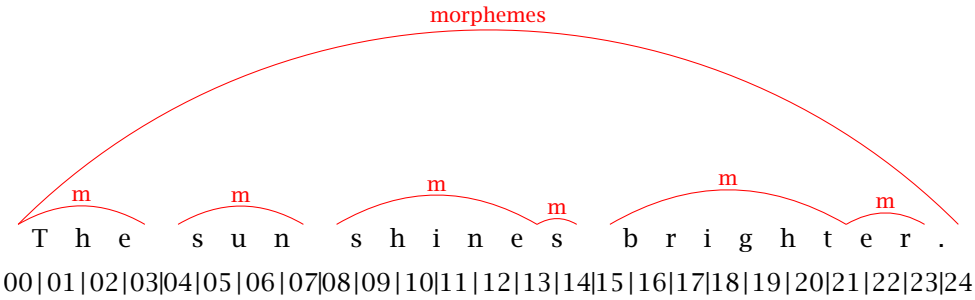


Abbildung 13.1: Grafische Repräsentation der Morphem-Annotation

Listing 13.4: Silben-Annotation

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <syllables>
3  <s>The</s>
4  <s>sun</s>
5  <s>shines</s>
6  <s>brigh</s>
7  <s>ter</s>.
8 </syllables>
```

Jeweils eine einzelne Annotationsebene wird pro Datei gespeichert (z. B. als Ergebnis der Verarbeitung durch eine linguistische Ressource). Eine grafische Repräsentation der Spannen über die Primärdaten der jeweiligen Elemente wird in den Abbildungen 13.1 und 13.2 sichtlich. Eine Kombination beider Annotationsebenen in einer einzelnen Inline-Annotation ist nicht möglich, da sich beim Wort „brighter“ Silben- und Morphemgrenzen überlappen (vgl. Listing 13.5 und Abbildung 13.3 auf der nächsten Seite). Sofern beide Annotationsebenen in separaten Dateien gespeichert werden, treten keine Probleme auf. Dem steht entgegen, dass bei einer solchen mehrfachen Speicherung neben der Redundanz der Primärdaten auch die Gefahr der Verfälschung und unbeabsichtigten Änderung der auszuzeichnenden Daten gegeben ist, z. B. durch Einfügen von Leerzeichen, Umbrüchen, etc.

Eine Lösung für diese Problematik bietet die Standoff-Auszeichnung, da hier nur lesend auf die Primärdaten zugegriffen wird. Allerdings erschwert die klassische Standoff-

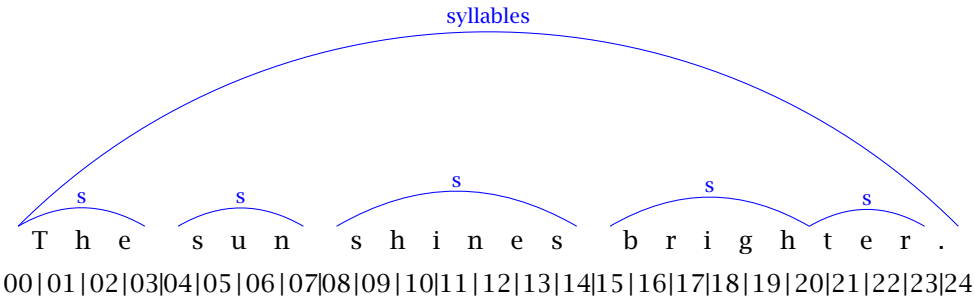


Abbildung 13.2: Grafische Repräsentation der Silben-Annotation

Listing 13.5: Nicht-wohlgeformte kombinierte Instanz beider Ebenen

```
<morphemes>
 <syllables>
  <m>
   <s>The</s>
  </m>
  <m>
   <s>sun</s>
  </m>
  <m>
   <s>shines</s>
  </m>
  <m>
   <s>brigh</s><s>t</m><m>er</s>
  </m>.
 </syllables>
</morphemes>
```

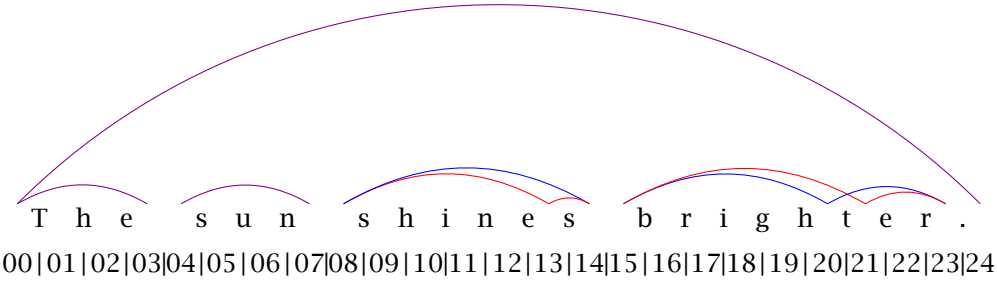


Abbildung 13.3: Grafische Repräsentation der überlappenden Annotationsebenen

Annotation, in der üblicherweise Primärdaten und jeweils eine einzelne Annotationsebene in separaten Dateien gespeichert werden (vgl. beispielsweise die Ausführungen zu LAF in Kapitel 9), die Analyse der Beziehungen zwischen Elementen verschiedener Annotationsebenen. Aus den genannten Gründen versucht XSTANDOFF die in Abschnitt 3.3.2.3 angeführten Vorzüge der Standoff-Annotation zu nutzen, ohne deren Nachteile in Kauf zu nehmen.

13.2 XStandoff – der grundlegende Aufbau

Eine XSTANDOFF-Instanz besteht im Wesentlichen aus den Primärdaten, deren Segmentierung und einer beliebigen Anzahl von dazugehörigen Annotationen – in einer Datei. Zur differenzierten Speicherung mehrerer Annotationsebenen mit möglicherweise gleich lautenden Elementnamen (Generic Identifier) werden diese innerhalb der Instanz physisch und mittels XML NAMESPACES logisch separiert. Elemente aus dem Namensraum `http://www.xstandoff.net/2009/xstandoff/1.1` gehören dem *Base Layer* an, der alle genannten Komponenten umfasst. Per Konvention wird für diesen Namensraum das Präfix `xsf` verwendet, andere Präfixe sind aber gemäß der XML NAMESPACE-Spezifikation möglich.

Das Wurzelelement einer XSTANDOFF-Instanz ist entweder das `corpus`- oder `cor-`

pusData-Element. Jedes corpusData-Element wird durch ein ID-Attribut eindeutig identifizierbar. Hierzu wird das generische xml:id-Attribut aus dem Namensraum http://www.w3.org/XML/1998/namespace genutzt, um unnötige Doppeldeklarationen zu vermeiden und die Wiederverwendbarkeit zu erleichtern. Listing 13.6 zeigt eine (nicht-valide) minimale XSTANDOFF-Instanz.

Anmerkung Aus Gründen der Lesbarkeit werden im Folgenden in den XSTANDOFF-Beispiel-Listings die Angaben zu den XML-Schemata nicht aufgeführt.

Listing 13.6: Aufbau einer XStandoff-Instanz

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence">
3  <!-- [...] -->
4 </xsf:corpusData>
```

13.2.1 Speicherung der Primärdaten

Ein corpusData-Element besitzt mindestens ein primaryData-Kindelement, das entweder die Primärdaten direkt speichert (unterhalb des Elements textualContent) oder auf eine externe Primärdatendatei verweist (mittels primaryDataRef-Element). Die Listings 13.7 und 13.8 zeigen beide Möglichkeiten.

Listing 13.7: Einbettung der Primärdaten in einer XStandoff-Instanz

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence">
3  <xsf:primaryData start="0" end="24" xml:lang="en" xml:space="preserve">
4   <xsf:textualContent>The sun shines brighter.</xsf:textualContent>
5  </xsf:primaryData>
6 </xsf:corpusData>
```

Listing 13.8: Referenz auf externe Primärdaten in einer XStandoff-Instanz

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence">
3  <xsf:primaryData start="0" end="24" xml:lang="en">
4   <xsf:primaryDataRef uri="sentence.txt" encoding="utf-8" mimeType="text/plain"/>
5  </xsf:primaryData>
6 </xsf:corpusData>
```

Im Falle von multi-modalen Annotationen kann es notwendig sein, mehr als eine Primärdatendatei als Annotationsbasis heranzuziehen (z. B. eine Videodatei nebst Transkription, vgl. Abschnitt 15.2). In diesem Fall werden mehrere Vorkommen von primaryData eingesetzt, von denen genau eines über das optionale role-Attribut als „Master“ bezeichnet wird.[1] Die so gekennzeichnete Primärdatendatei stellt die

[1] Die Einschränkung, dass bei mehreren primaryData-Elementen nur eines ein Attribut role mit dem Wert *master* haben kann, wird im XSTANDOFF definierenden Schema über eingebettete SCHEMATRON-Business Rules bzw. in der mittels XSD 1.1 realisierten Entwicklerversion über xs:assert-Elemente sichergestellt (vgl. auch Abschnitt 13.4.1).

Referenz für die Segmentierung (vgl. Abschnitt 13.2.2) dar, andere Primärdaten können ausgehend von dieser Basis auch nur Teile (z. B. eine partielle Transkription) der Spanne an Zeichen bzw. Dauer umfassen (hierzu kann mittels `offset`-Attribut ein entsprechender Versatz angegeben werden).

Die Attribute `start` und `end` speichern den Wert der ersten bzw. letzten Segmentierungseinheit (Zeichen, sofern es sich um einen Text handelt) des Primärdatums. Dabei wird jedes Zeichen inkl. Weißraum-Zeichen (Leerzeichen, Umbrüche, Tabstops etc.) gezählt, empfehlenswert ist daher eine vorherige Normalisierung der Primärdaten in Bezug auf solche Zeichen (für die Verarbeitung von Weißraum-Zeichen im Rahmen des XStandoff-Toolkits vgl. Kapitel 14). Es besteht die Möglichkeit, mittels des optionalen Elements `checksum` eine Prüfsumme für die Primärdaten zu speichern (z. B. mittels MD5 oder eines ähnlichen Algorithmus'), die gewährleistet, dass externe Ressourcen auf dem gleichen Eingabetext arbeiten.

Die Möglichkeit, die Primärdaten sowohl intern als auch extern speichern zu können, birgt einige Vorteile: Die erste Variante erlaubt eine *All-in-one*-Lösung, die ohne weitere Dateien (mit Ausnahme der entsprechenden Dokumentgrammatiken) funktioniert, was z. B. als Austauschformat sinnvoll sein kann. Die Alternative ist dann ratsam, wenn die Primärdaten nicht digital vorliegen, aufgrund rechtlicher Einschränkungen nicht zusammen mit der Annotation weitergegeben werden dürfen, sehr umfangreich sind, oder eine andere Kodierung als die XStandoff-Instanz aufweisen. Ebenso können mehrere Dateien als Primärdaten genutzt werden (z. B. mehrere Video- bzw. Audiodateien, nebst Transkription im Zuge einer multimodalen Annotation, vgl. Abschnitt 15.2).

Im Beispiel nicht sichtbar ist die Möglichkeit mittels des `unit`-Attributs eine andere Segmentierungseinheit (Standardwert ist *char* für einzelne Zeichen bei der Verwendung textueller Primärdaten) anzugeben. Neben dem bereits erwähnten *char* sind die weiteren erlaubten Werte *bytes*, *microseconds*, *milliseconds*, *seconds*, *minutes*, *hours*, *days*, *weeks*, *months*, *years*, *palFrames* bzw. *ntscFrames* (für Einzelbilder nach dem PAL- bzw. NTSC-Video-Standard) oder *other*.[2] Aktuell sind diese Werte in der Dokumentgrammatik für XStandoff fest hinterlegt, zukünftige Versionen könnten auch von standardisierten Segmentierungsschemata profitieren, wie sie hoffentlich in der finalen Version des Linguistic Annotation Framework enthalten sein werden.

13.2.2 Segmentierung

Ausgehend von den beiden Beispielannotationen aus den Listings 13.3 und 13.4 auf Seite 245 können die Positionen der Elementgrenzen bestimmt werden. Diese werden in XStandoff als Segmente (`segment`-Elemente) gespeichert. Jedes Segment ist durch den dokumentweit eindeutigen Wert des obligatorischen `xml:id`-Attributs identifizierbar und besitzt Start- und Endkoordinaten in Form numerischer Werte (in der Dokument-

[2] Im Phase-Alternation-Line-Verfahren (PAL) gibt es zwei Wiederholfrequenzen: 25 und 29,97 Bilder/-Sekunde. Da das National Television Systems Committee-Verfahren (NTSC) allerdings ebenfalls mit einer Bildwiederholrate von 29,97 Bildern/Sekunde arbeitet, reichen die beiden genannten Werte zur Unterscheidung aus. Bei Verwendung von *other* ist eine erklärende Angabe zur Segmentierungseinheit in den Metadaten der Instanz (vgl. Abschnitt 13.3.1) erforderlich.

Tabelle 13.1: Segmentgrenzen der Silben-Annotation

Element	Startposition	Endposition
syllables/morphemes	0	24
s/m	0	3
s/m	4	7
m	8	13
s	8	14
s	15	20
m	15	21
s	20	23
m	21	23

grammatik als vom Datentyp `xs:nonNegativeInteger` deklariert).[3] Es ist daher als äquivalent zum `token`-Element des MORPHO-SYNTACTIC ANNOTATION FRAMEWORK (vgl. Kapitel 10 anzusehen). Über ein optionales `type`-Attribut kann der Segmenttyp genauer festgelegt werden. Der Standardwert ist *char* (zur Festlegung, dass ein Segment eine Zeichenkette begrenzt), weitere mögliche Werte sind *empty* (für leere Elemente), *ws* (für Weißraum-Zeichen), *pun* (für Interpunktionszeichen), *dur* (Zeitspannen) oder *seg* (diskontinuierliche und virtuelle Elemente, vgl. Abschnitt 13.3.2).[4] Das optionale Attribut `content` kann zusätzlich den durch dieses Segment umspannten Bereich der Primärdaten beinhalten. Ein solches Vorgehen kann dann sinnvoll sein, wenn die Primärdaten extern gespeichert und mittels `primaryDataRef` referenziert werden, um eine bessere Lesbarkeit zu ermöglichen. Zur besonderen Behandlung Unicode-kodierter Texte kann über das optionale `unicodeNormalizationForm`-Attribut der verarbeitenden Software mitgeteilt werden, in welcher der vier möglichen Normalisierungsformen (NFD, NFC, NFKD und NFKC) zusammengesetzte Zeichen zu verarbeiten sind (vgl. Unicode 6.0.0, Anhang 15, „Unicode Normalization Forms“). Das `segment`-Element kann als leeres Element genutzt werden, um als Anker für spätere Annotationen zu dienen, oder Meta-Informationen beinhalten (ein Beispiel dafür zeigt Listing 13.17 auf Seite 260).

Jedes Element einer Annotationsebene resultiert in einem Segment, für die beiden Beispielannotation ergeben sich damit die in Tabelle 13.1 gezeigten Segmente. Dabei zeigt sich, dass mehrere Elemente beider Annotationsebenen die gleichen Segmentgrenzen haben (u. a. natürlich das Wurzelelement, das den gesamten Text umspannt). Diesen Umstand macht sich XSTANDOFF zu Nutze: Segmente, mit gleicher Start- und Endposition werden nur einmal erstellt (analog zu der Tokenisierung in MAF). In einer XSTANDOFF-Instanz werden diese Segmente unterhalb des Elements `segmentation` aufgeführt (vgl. Listing 13.9).

Leere Elemente erhalten sowohl bei Start- als auch Endposition den gleichen Wert (im Beispiel nicht enthalten). Im Gegensatz zur Segmentierung des LINGUISTIC ANNOTATION FRAMEWORK (vgl. Abschnitt 9.1) können Segmentierungen auch überlappend angelegt werden.

[3] Aufgrund der mangelhaften Unterstützung von XPOINTER (vgl. die Ausführungen in Abschnitt 7.4) wurde davon Abstand genommen und dieser robusteren Lösung der Vorzug gegeben.
[4] Zu beachten ist, dass diese Angabe sich nicht auf die Segmentierungseinheit bezieht.

Listing 13.9: Segmente in der XStandoff-Instanz

```
<xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence">
 <xsf:primaryData start="0" end="24">
  <xsf:primaryDataRef uri="sentence.txt" encoding="utf-8" mimeType="text/plain"/>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
  <xsf:segment xml:id="seg2" type="char" start="0" end="3"/>
  <xsf:segment xml:id="seg3" type="char" start="4" end="7"/>
  <xsf:segment xml:id="seg4" type="char" start="8" end="14"/>
  <xsf:segment xml:id="seg5" type="char" start="8" end="13"/>
  <xsf:segment xml:id="seg6" type="char" start="13" end="14"/>
  <xsf:segment xml:id="seg7" type="char" start="15" end="21"/>
  <xsf:segment xml:id="seg8" type="char" start="15" end="20"/>
  <xsf:segment xml:id="seg9" type="char" start="20" end="23"/>
  <xsf:segment xml:id="seg10" type="char" start="21" end="23"/>
 </xsf:segmentation>
</xsf:corpusData>
```

13.2.3 Umwandeln der Annotationsebenen

Sind die Segmente berechnet, können diese mit den Annotationen in Bezug gesetzt werden. Dazu werden die ursprünglichen Annotationsebenen transformiert. Grundgedanke dabei ist, so wenig Änderungen wie möglich vorzunehmen. Das hat zwei Gründe: Zum einen soll generell möglichst viel der ursprünglichen Auszeichnung erhalten bleiben, zum anderen sind die Beziehungen zwischen Knoten wie z. B. die Eltern-Kind- bzw. Geschwister-Relation XML-inhärent kodiert – es ist daher nach Ansicht des Verfassers unnötig, sie explizit mittels Hilfskonstruktionen wie einer Structure List zu rekodieren, zumal sämtliche XML-unterstützende Software diese Relationen ausnutzen kann.

Jede Annotationsebene wird wie folgt umgewandelt:

- Alle Elemente (auch leere) erhalten ein Attribut `segment` aus dem XStandoff-Namensraum, das zur Identifikation des im vorherigen Schritt angelegten Segments dient, die das Element in der Ursprungsannotation annotiert hat.
- Elemente, die andere Elemente und Zeichenketten enthalten können (mixed content), werden zu reinen Containerelementen abgeändert.
- Datenelemente werden zu leeren Elementen umgewandelt.

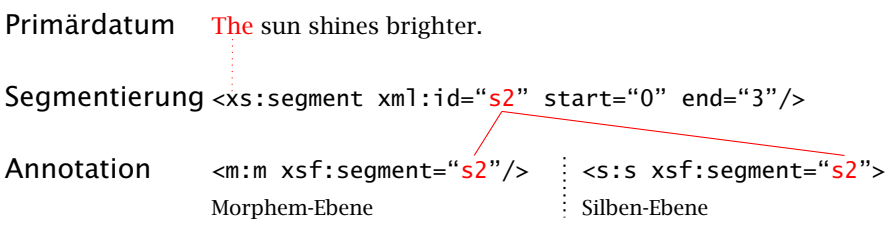
Attribute bleiben unverändert ihren jeweiligen Elementen zugeordnet. Um die Annotationsebenen voneinander zu trennen, werden XML Namespaces genutzt. Sind die ursprünglichen Inline-Auszeichnungen bereits einem solchen zugeordnet, wird dieser übernommen, ansonsten wird ein neuer Namensraum der Form `http://www.xstandoff.net/<X>` erzeugt.

Die transformierte Annotationsebene wird unterhalb des `annotation`-Elements gespeichert. Den Ausführungen in Abschnitt 3.5 folgend, unterscheidet XStandoff zwischen einer konzeptuellen Ebene (Level, Element `level`) und der konkreten Notation (Layer, Element `layer`). Dabei gilt das bereits Gesagte: Unterhalb von `annotation` sind mehrere Vorkommen von `level` erlaubt (um unterschiedliche Betrachtungsweisen

auf ein Primärdatum realisieren zu können), die jeweils wiederum mehrere `layer`-Kindelemente haben können (z. B. um verschiedene POS-Parsebäume miteinander vergleichen zu können). Ein `level` kann über das optionale Attribut `dcrCategory` (vom Datentyp `xs:anyURI`) auf eine standardisierte Datenkategorie (z. B. aus ISOCAT, vgl. die Ausführungen in Kapitel 8) verweisen und so näher spezifiziert werden. Diese Referenz ist auch und gerade dann sinnvoll, sofern die importierten Annotationsebenen nicht über eine solche Verknüpfung verfügen. Für die einzelnen Elemente einer importierten Annotationsebene ist das gleiche Attribut ebenfalls optional nutzbar, so dass die Gegenüberstellung von lokalem Vokabular zu standardisierter Datenkategorie möglich ist. In diesem Fall kann eine XSTANDOFF-Instanz einen informatorischen Mehrwert gegenüber der ursprünglichen Inline-Annotation aufweisen und könnte – in einem späteren Schritt – sogar zur Anreicherung bzw. Erweiterung einer bestehenden Datenkategorie in ISOCAT herangezogen werden.

Das Element `layer` als Kind von `level` ist als *Wrapper*-Element deklariert, das die zu importierende Repräsentation einer Annotationsebene umschließt. Intern ist es als Wildcard deklariert, die Teilbäume aus einem anderen Namensraum als den von XSTANDOFF als Kinder erlaubt (vgl. Abschnitt 3.4.4.3 und Walmsley 2002, S. 278ff.). Damit erlaubt XSTANDOFF die Kombination verschiedener Auszeichnungssprachen mitsamt deren spezifischen Merkmalen, ein Vorteil der Standoff-Annotation, der auch von Ule und Hinrichs (2004, S. 231) herausgestellt wird: „Das Konzept der *Standoff-Annotation* lässt sich natürlich auch ebenso einsetzen, um die Vorteile einzelner Annotationsformate zu kombinieren [...]“

Die Verbindung zwischen Auszeichnung und der ursprünglich davon umschlossenen Textspanne in den Primärdaten (die ja nun durch ein `segment`-Element denotiert ist), wird über eine `ID/IDREF`-Relation (also ein Integritätsmerkmal) hergestellt: Über das neu hinzugefügte `segment`-Attribut wird der Wert des `xml:id`-Attribut des entsprechenden `segment`-Elements referenziert; Abbildung 13.4 verdeutlicht dies graphisch.


Primärdatum
The sun shines brighter.
Segmentierung

Annotation
<m:m xsf:segment="s2"/>
<s:s xsf:segment="s2">
Morphem-Ebene
Silben-Ebene

Abbildung 13.4: Nutzung des `ID/IDREF`-Mechanismus in XStandoff

Listing 13.10 zeigt die XSTANDOFF-Instanz inkl. Verweis auf externe Primärdaten, aller Segmente und beider Annotationsebenen.

Listing 13.10: Die vollständige XStandoff-Instanz

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence">
3  <xsf:primaryData start="0" end="24">
4   <xsf:primaryDataRef uri="sentence.txt" encoding="utf-8" mimeType="text/plain"/>
5  </xsf:primaryData>
6  <xsf:segmentation>
```

```
 <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
 <xsf:segment xml:id="seg2" type="char" start="0" end="3"/>
 <xsf:segment xml:id="seg3" type="char" start="4" end="7"/>
 <xsf:segment xml:id="seg4" type="char" start="8" end="14"/>
 <xsf:segment xml:id="seg5" type="char" start="8" end="13"/>
 <xsf:segment xml:id="seg6" type="char" start="13" end="14"/>
 <xsf:segment xml:id="seg7" type="char" start="15" end="21"/>
 <xsf:segment xml:id="seg8" type="char" start="15" end="20"/>
 <xsf:segment xml:id="seg9" type="char" start="20" end="23"/>
 <xsf:segment xml:id="seg10" type="char" start="21" end="23"/>
</xsf:segmentation>
<xsf:annotation>
 <xsf:level xml:id="l_morph">
  <xsf:layer xmlns:m="http://www.xstandoff.net/morph" priority="0">
   <m:morphemes xsf:segment="seg1">
    <m:m xsf:segment="seg2"/>
    <m:m xsf:segment="seg3"/>
    <m:m xsf:segment="seg5"/>
    <m:m xsf:segment="seg6"/>
    <m:m xsf:segment="seg7"/>
    <m:m xsf:segment="seg10"/>
   </m:morphemes>
  </xsf:layer>
 </xsf:level>
 <xsf:level xml:id="l_syll">
  <xsf:layer xmlns:s="http://www.xstandoff.net/syll" priority="1">
   <s:syllables xsf:segment="seg1">
    <s:s xsf:segment="seg2"/>
    <s:s xsf:segment="seg3"/>
    <s:s xsf:segment="seg4"/>
    <s:s xsf:segment="seg8"/>
    <s:s xsf:segment="seg9"/>
   </s:syllables>
  </xsf:layer>
 </xsf:level>
</xsf:annotation>
</xsf:corpusData>
```

Abbildung 13.5 auf der nächsten Seite zeigt abschließend den Aufbau einer XStandoff-Instanz in einer Übersicht.

Nachdem damit grundlegende Komponenten des Metaformats vorgestellt sind, werden die folgenden Abschnitte näher auf einzelne Aspekte von XStandoff eingehen. Es sollte an dieser Stelle deutlich geworden sein, dass XStandoff im Gegensatz zu anderen Standoff-Formaten nicht zwingend die Verwendung mehrerer Dateien vorschreibt, sondern einen „All-in-One“-Ansatz propagiert.

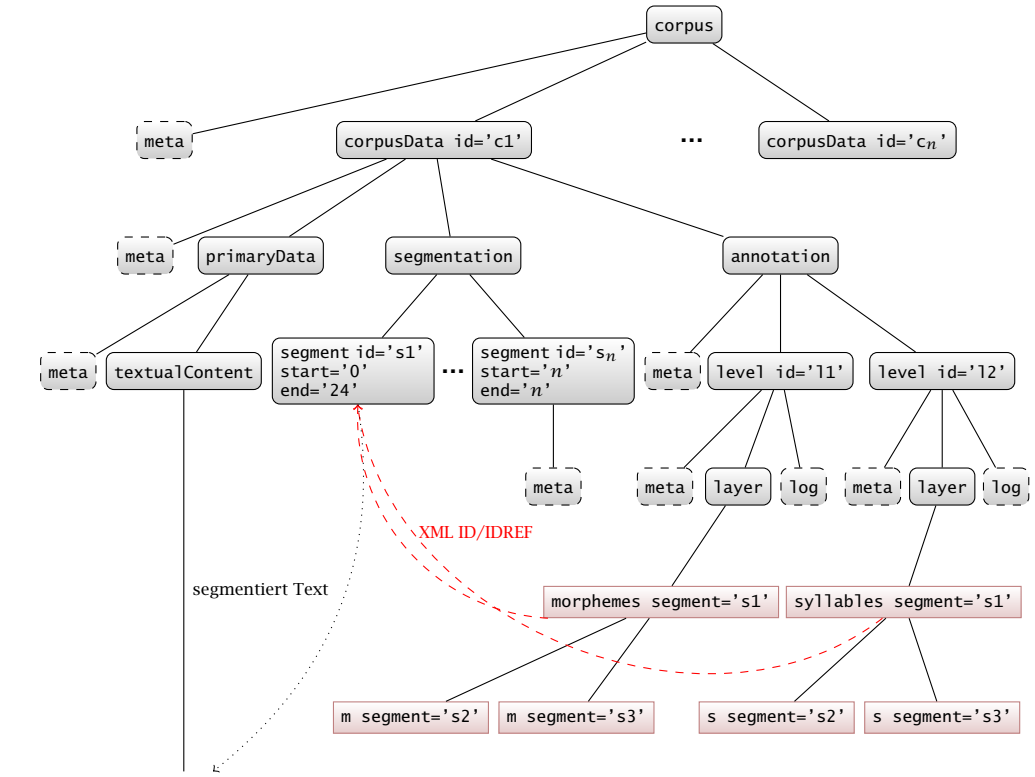


Abbildung 13.5: Übersicht über den Aufbau einer XStandoff-Instanz

13.3 Weitere Merkmale

Zu den weiteren Merkmalen von XStandoff zählt die Verwendung von Metadaten, die Unterstützung von diskontinuierlichen Annotationseinheiten, die Möglichkeit, eine Bearbeitungshistorie intern zu speichern und der `all`-Layer zur Zusammenfassung von Teilbäumen, die auf allen Annotationsebenen präsent sind.

13.3.1 Metadaten

Metadaten sind ein essentieller Bestandteil von Korpora: Sie ermöglichen Navigations- und Retrievalstrategien und beschreiben Korpusteile (und den gesamten Korpus) maschinenlesbar (vgl. Kapitel 12). Daher haben sie auch in XStandoff einen hohen Wert und können an verschiedenen Stellen in einer XStandoff-Instanz eingefügt werden. Vorkommen des optionalen Elements `meta` sind direkt unterhalb der beiden möglichen Wurzelelemente `corpus` und `corpusData`, unterhalb von `primaryData`, `segmentation`, `annotation`, `level` und `layer` möglich. Darüber hinaus können auch Elemente des Typs `segment` Meta-Angaben beinhalten. Wie bei den Annotationsebenen auch gibt es in dieser Hinsicht keine Vorgaben durch XStandoff, d. h., es können jegliche Metadatenformate zum Einsatz kommen – daher gelten die in Kapitel 12 gemachten

Empfehlungen. Listing 13.11 zeigt die exemplarische Einbindung von Metadaten der *Open Language Archives Community* (vgl. Abschnitt 12.2) für die bereits bekannte Instanz.

Listing 13.11: Metadaten

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence"
    xsfVersion="1.1">
 <xsf:meta xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
  xmlns="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/">
  <olac:olac>
   <creator>Maik Stührenberg</creator>
   <date>2011-05-19</date>
   <description>Sentence "The sun shines brighter" annotated on both morpheme and syllable
    levels.</description>
  </olac:olac>
 </xsf:meta>
 <xsf:primaryData start="0" end="24">
  <xsf:primaryDataRef uri="sentence.txt" encoding="utf-8" mimeType="text/plain"/>
 </xsf:primaryData>
 <!-- [...] -->
</xsf:corpusData>
```

Ebenso lassen sich Metadaten unterhalb einer konzeptuellen Annotationsebene (`level`) oder unter einer konkreten Serialisierung (`layer`) speichern, z. B., um den Ersteller der Auszeichnung zu nennen (hierzu gibt es mittels des `creator`-Attributs unterhalb von `layer` eine gesonderte Möglichkeit). Im XSTANDOFF definierenden XML-Schema (vgl. Anhang) ist für das Wrapper-Element `meta` das Attribut `processContents` mit dem Wert *lax* belegt, d. h., der Schema-Prozessor, der zur Validierung der Instanz herangezogen wird, validiert nur solche Elemente, für die er eine Deklaration (in Form eines XML-Schemas) vorfindet und weist entsprechend ungültige Elementinstanzen mittels einer Fehlermeldung zurück. Elemente, für die keine Deklaration vorhanden ist, erzeugen keine Fehlermeldung (vgl. Walmsley 2002, S. 279), dementsprechend sind auch schwach strukturierte Metadaten möglich, wie Listing 13.12 ausschnittsweise zeigt.[5]

Listing 13.12: Schwach strukturierte Metadaten

```
<!-- [...] -->
<xsf:meta>
 <header xmlns="http://www.xstandoff.net/simpleMeta">
  Autor: Maik Stührenberg
  Datum: 19.05.2011
 </header>
</xsf:meta>
<!-- [...] -->
```

[5] Zu beachten ist, dass zumindest ein Wurzelelement des unterhalb von `meta` beginnenden Teilbaums vorhanden sein muss, das nicht aus dem XSTANDOFF-Namensraum stammt (im Beispiel `header`).

13.3.2 Diskontinuierliche und virtuelle Annotationseinheiten

Das bereits in Abschnitt 3.3.2 gezeigte Beispiel-Listing 3.8 auf Seite 41 der ersten Sätze aus „Alice in Wonderland“ kann als Beispiel für diskontinuierliche Annotationseinheiten dienen, die von XStandoff unterstützt werden.

Die beiden getrennten q-Fragmente zeichnen zwei Teile einer größeren, zusammenhängenden Phrase aus, die inline nicht ohne Überlappungen annotiert werden kann. XStandoff ermöglicht diskontinuierliche Annotationseinheiten durch die Erzeugung eines `segment`-Elements, dass sich derart von gewöhnlichen Segmenten unterscheidet, dass es nicht über die Attribute `start` und `end` verfügt, sondern über das `segments`-Attribut, das intern vom Token-Typ `xs:IDREFS` deklariert ist, also Referenzen zu mehreren anderen durch einen eindeutigen Bezeichner identifizierbaren Elementknoten herstellt (in diesem Fall zu anderen Segmenten). Zusätzlich wird das `type`-Attribut auf den Wert *seg* gesetzt und das `mode`-Attribut erhält den Wert *disjoint* (Standardwert ist hier *continuous*). Listing 13.13 zeigt die entsprechende Serialisierung in XStandoff.

Listing 13.13: Diskontinuierliche Elemente in XStandoff

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="alice">
3   <xsf:primaryData start="0" end="302">
4    <xsf:primaryDataRef uri="alice.txt" encoding="utf-8" mimeType="text/plain"/>
5   </xsf:primaryData>
6   <xsf:segmentation>
7    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
8    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
9    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
10   <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
11  </xsf:segmentation>
12  <xsf:annotation>
13   <xsf:level xml:id="alice-log">
14    <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log" priority="0">
15     <log:text xsf:segment="seg1">
16      <log:p xsf:segment="seg1">
17       <log:q xsf:segment="seg4"/>
18      </log:p>
19     </log:text>
20    </xsf:layer>
21   </xsf:level>
22  </xsf:annotation>
23 </xsf:corpusData>
```

Das Segment mit dem Bezeichner *seg4* in Zeile 10 besteht aus den beiden Segmenten *seg2* und *seg3* (Zeilen 8 und 9), wobei die Endposition von *seg2* ungleich der Startposition von *seg3* ist (auch unter Berücksichtigung von Weißraum-Zeichen), weshalb das `mode`-Attribut entsprechend auf den Wert *disjoint* zu setzen ist.

Dieses Merkmal von XStandoff kann auch dazu eingesetzt werden, die in GODDAGs und der Metasprache TexMECS realisierbare Unterscheidung von Dominanz und Einschluss (vgl. Abschnitt 3.3.3) abzubilden. Dominanz wird weiterhin durch die Mutter-Kindbeziehung zwischen zwei Knoten ausgedrückt, beim obigen Beispiel aus „Alice in Wonderland“ gibt es dabei drei mögliche Varianten: Das (diskontinuierliche, virtuelle) `q`-Element wird durch `p` dominiert, die beiden `q`-Fragmente werden durch `p` dominiert,

oder p und q sind vollkommen unabhängig voneinander. Die erste Variante ist bereits in Listing 13.13 realisiert, Listing 13.14 demonstriert die zweite, Listing 13.15 die letzte. Zu beachten ist, dass der Einschluss weiterhin durch die Zeichenposition der Segmente gewährleistet wird.

Listing 13.14: Dominanzbeziehung: p dominiert die q-Fragmente

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="al">
 <!-- [...] -->
 <xsf:annotation>
  <xsf:level xml:id="alice-log">
   <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log" priority="0">
    <log:text xsf:segment="seg1">
     <log:p xsf:segment="seg1">
      <log:q xsf:segment="seg2"/>
      <log:q xsf:segment="seg3"/>
     </log:p>
     <log:q xsf:segment="seg4"/>
    </log:text>
   </xsf:layer>
  </xsf:level>
 </xsf:annotation>
</xsf:corpusData>
```

Listing 13.15: Dominanzbeziehung: p und q sind unabhängig

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="al">
 <!-- [...] -->
 <xsf:annotation>
  <xsf:level xml:id="alice-log">
   <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log" priority="0">
    <log:text xsf:segment="seg1">
     <log:p xsf:segment="seg1"/>
     <log:q xsf:segment="seg4"/>
    </log:text>
   </xsf:layer>
  </xsf:level>
 </xsf:annotation>
</xsf:corpusData>
```

Die durch dieses Merkmal realisierbaren virtuellen Elemente lassen sich auch zur Umordnung von Teilbäumen nutzen. Auch hier geht das Beispiel auf andere Arbeiten zurück, in diesem Fall auf die TEI GUIDELINES (P5 1.9.1, Kapitel 16, „Linking, Segmentation, and Alignment“). Ausgangslage ist der folgende Dialog zwischen Huey, Dewey, und Louie (im Deutschen besser bekannt als Tick, Trick und Track), die versuchen einen Haiku zu rezitieren, die Zeilen aber in der falschen Reihenfolge hervorbringen. Listing 13.16 auf der nächsten Seite zeigt eine TEI-Annotation, die Elemente sp (*speech*, wörtliche Rede), p (*paragraph*, Absatz), q (durch Anführungszeichen abgegrenzter Textteil), l (*line*, Zeile) nutzend.

In den TEI GUIDELINES existiert mit dem join-Element eine Möglichkeit, die Zeilen des Haikus als virtuelles Element neu anzuordnen (Zeile 27, vgl. auch Abschnitt 5.5). Diese

Listing 13.16: TEI-Annotation des Haiku-Dialogs

```
<?xml version="1.0" encoding="UTF-8"?>
<text xmlns="http://www.tei-c.org/ns/1.0">
 <body>
  <sp who="Hughie">
   <p>How does it go? <q>
     <l xml:id="frog-X1">da-da-da</l>
     <l xml:id="frog-L2">gets a new frog</l>
     <l>...</l>
    </q>
   </p>
  </sp>
  <sp who="Louie">
   <p>
    <q>
     <l xml:id="frog-L1">When the old pond</l>
     <l>...</l>
    </q>
   </p>
  </sp>
  <sp who="Dewey">
   <p>
    <q>
     <l>...</l>
     <l xml:id="frog-L3">It 's a new pond.</l>
    </q>
   </p>
   <join targets="#frog-L1 #frog-L2 #frog-L3" result="lg" scope="root" type="haiku" />
  </sp>
 </body>
</text>
```

Möglichkeit besteht auch in XStandoff, da sich die TEI-Annotationsebene natürlich unterhalb eines `layer`-Elements in einer XStandoff-Instanz einbetten lässt – womit aber kein Mehrwert zur alleinigen TEI-Instanz erzielt wäre. Es gibt darüber hinaus aber noch eine XStandoff-eigene Variante, die sich zu Nutze macht, dass unterhalb eines `segment`-Elements Metadaten erlaubt sind. Listing 13.17 auf Seite 260 zeigt eine mögliche Realisierung.

Neben dem `join`-Element (analog zur TEI-Lösung) werden über zusätzliche Metadaten weitere Angaben zum virtuellen Element. Zusätzlich wird das virtuelle Element zweifach erzeugt (über das diskontinuierliche Segment *seg13*). Ebenfalls sichtbar wird, dass XStandoff mehrfache Metadatenknoten erlaubt, im Listing 13.17 auf Seite 260 zunächst in Zeile 11 das `join`-Element aus dem TEI-Namenraum, und zusätzlich in Zeile 13 das `olac`-Element aus dem Namensraum der *Open Language Archives Community Metadata* (vgl. Abschnitt 12.2).

13.3.3 Der `all`-Layer

Eine der Unterschiede zwischen SGF und XStandoff ist die Einführung des `all`-Layers. Hintergrund dieser besonderen Annotationsebene ist die weitere Komprimierung der XStandoff-Instanz durch Zusammenfassen von Elementen, die mehrfach (d. h., in verschiedenen Layern) enthalten sind. Diese lassen sich in den `all`-Layer überführen, der

dem XML Namespace `http://www.xstandoff.net/2009/all` angehört. Der eigene Namensraum ist aus zwei Gründen notwendig: Aus technischer Sicht erfordert das XStandoff definierende Schema, dass der Teilbaum unterhalb des `layer`-Elements einem anderen Namensraum angehört. Zum anderen ist es aus semantischer Sicht folgerichtig, diese „virtuelle“ Annotationsebene einem festgelegten Namensraum zuzuordnen. Der `all`-Layer nimmt auf Wunsch alle Elemente auf, die in allen konkreten Annotationsebenen mit dem gleichen Namen und den gleichen Segmentgrenzen vorhanden sind. Dabei ist zu beachten, dass dieser virtuelle Layer keinerlei Aussagen über die Semantik der Auszeichnungen macht (vgl. hier auch die allgemeine Diskussion um Semantik von Element- und Attributnamen in Kapitel 8). Ein Beispiel für die Verwendung des `all`-Layers ist in Listing 15.2 auf Seite 277 aufgeführt.

13.3.4 Änderungshistorie: das Log

Ein gerne übersehener Aspekt der Erstellung linguistischer Korpora ist der der Dokumentation einer Änderungshistorie des gesamten Korpus bzw. einzelner Einträge. Darunter fallen Änderungen wie das Hinzufügen von Annotationsebenen bzw. die Korrektur vorhandener Auszeichnungen. Oftmals werden Korpusdaten als abgeschlossene Instanzen veröffentlicht, so dass für nicht an der Erstellung beteiligten Nutzer nur die unannotierten Primärdaten und die vollständig annotierte Instanz einsehbar sind. Dabei können auch wie auch immer motivierte Änderungen von Relevanz sein, beispielsweise Korrekturen menschlicher Annotatoren. Als Serialisierungsformat des Web-basierten Annotationswerkzeugs Serengeti (vgl. Stührenberg, Goecke *et al.* 2007; Diewald *et al.* 2008; Poesio *et al.* 2011) verfügt bereits SGF über eine dokumentierte Änderungshistorie.[6] Während der Weiterentwicklung zu XStandoff wurde dieser überarbeitet und erweitert. Das optionale Element `log`, das unterhalb der Elemente `annotation`, `layer`, und `layer` auftreten kann, enthält neben Metadaten mindestens ein Vorkommen des `entry`-Elements. Dieses wiederum hat als mögliche Kindelemente `add`, `modify` oder `delete` (vgl. Abbildung 13.6).[7]

Werden Änderungen an den Annotationsebenen durchgeführt und sollen diese dokumentiert werden, werden diese unterhalb des jeweiligen `log`-Elements in Form des Elements `entry` durch die entsprechenden Kind-Elemente spezifiziert. Das Hinzufügen eines neuen Auszeichnungselements erfolgt durch demnach durch ein `add`-Element. Im Beispiel-Listing 13.18 wird die Silbenebene aus dem bekannten Beispiel so eingefügt. Eine Änderung an bestehenden Daten wird durch ein `modify` signalisiert. Dieses dient ähnlich wie `add` als Wrapper für den durch das `affectedItem`-Attribut referenzierten zu ändernden Teilbaums. Zu beachten ist dabei, dass `add` mit dem Wert *strict* des `processContents`-Attributs deklariert ist, während `modify` den Wert *skip* nutzt. Das ist notwendig, damit bei geänderten Teilbäumen `xs:ID`-Attribute mehrfach mit dem gleichen Wert genutzt werden können. Das Element `delete` ist nur dagegen ein lee-

[6] Eine öffentlich zugängliche Version von Serengeti ist unter `http://coli.lili.uni-bielefeld.de/serengeti/` zu finden, zuletzt abgerufen am 19.04.2012.

[7] Auch bei Nutzung des Logs sollten weiterhin menschen- und maschinenlesbare Metadaten *zusätzlich* die vollzogenen Änderungen an der Instanz dokumentieren, da das Log nur die automatisierbare Rückabwicklung bestimmter Aktionen erleichtert.

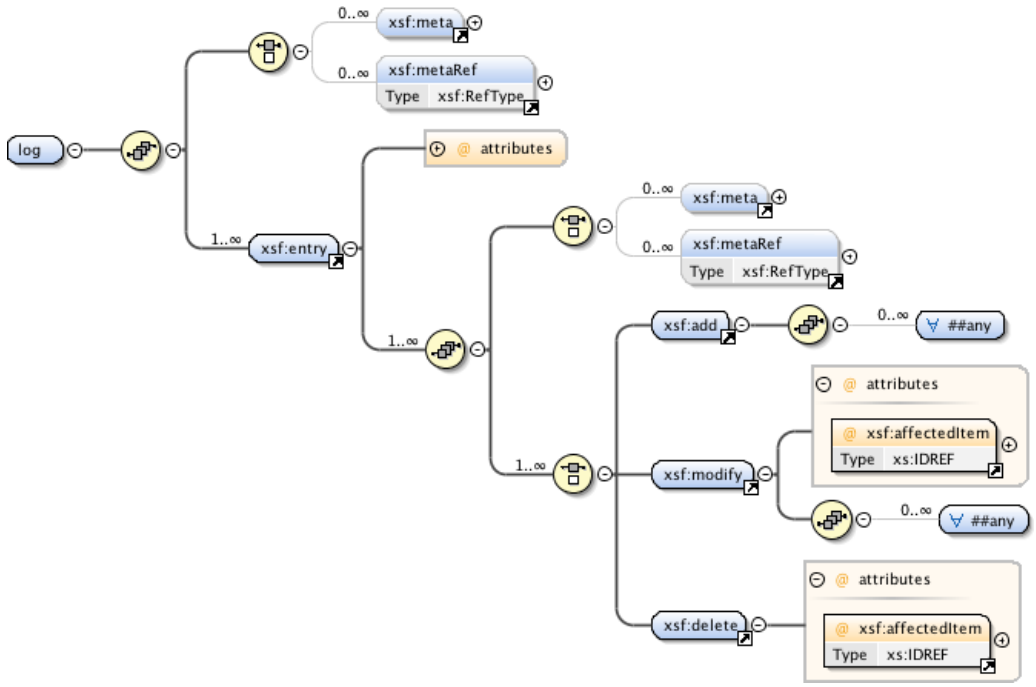


Abbildung 13.6: Grafische Darstellung des `log`-Elements

res Element, das über das Attribut `affectedItem` den Bezeichner des referenzierten Teilbaums speichert, der entfernt werden soll.

Die hiermit erzielte Speicherung der Änderungshistorie ist (obgleich Teil der aktuellen Produktivversion von XStandoff) als prototypisch anzusehen. Zum einen benötigt sie für die entsprechenden Elemente oder Teilbäume, deren Bearbeitung aufgezeichnet werden soll, eindeutige Bezeichner. Zum anderen ist zu beachten, dass keine Änderungen an den Segmenten durch das Log gespeichert werden. Diese Entscheidung wurde bewusst getroffen, da die Segmentierung häufiger Änderungen unterworfen werden kann (z. B. durch Hinzufügen einer neuen Ebene inkl. Neuberechnung und Umbenennung der `segment`-Elemente, vgl. Abschnitt 14). Des Weiteren ist ein solches Log sinnvollerweise mit Unterstützung einer geeigneten Software durchzuführen (auch in Kombination mit weiteren Applikations-spezifischen Metadaten), ein Beispiel dafür ist das bereits angesprochene Serengeti. Da in diesem Sinne dieser Teil von XStandoff den ansonsten postulierten universellen Einsatz der Spezifikation mindert, ist der Einsatz der Änderungshistorie auf begründete Einzelfälle beschränkt und die Dokumentgrammatik für das Log als separates XML-Schema ausgeführt, das per `xs:include` explizit eingebettet werden muss. Es zeigt aber – im Vergleich zur Verortung der Änderungshistorie in den Metadaten wie beispielsweise bei den TEI Guidelines (vgl. Abschnitt 5.3) – einen alternativen und detaillierteren Ansatz der Speicherung von Modifikationen auf.

Listing 13.17: Virtuelles Element mit neuer Anordnung in XStandoff

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="h">
 <xsf:primaryData start="0" end="87">
  <xsf:primaryDataRef uri="huey.txt"/>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" start="0" end="87"/>
  <!-- [...] -->
  <xsf:segment xml:id="seg13" segments="seg8 seg5 seg12" mode="disjoint">
   <xsf:meta>
    <join xmlns="http://www.tei-c.org/ns/1.0" targets="#frog-L1 #frog-L2 #frog-L3"
     result="lg" scope="root" type="haiku"/>
    <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
     xmlns="http://purl.org/dc/elements/1.1/">
     <creator>Maik Stührenberg</creator>
     <date>2011-05-19</date>
     <description>Erzeugtes join zur korrekten Anordnung der Zeilen.</description>
    </olac:olac>
   </xsf:meta>
  </xsf:segment>
 </xsf:segmentation>
 <xsf:annotation>
  <xsf:level xml:id="haiku-tei">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0">
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <sp who="Huey" xsf:segment="seg2">
       <p xsf:segment="seg2">
        <q xsf:segment="seg3">
         <l xml:id="frog-X1" xsf:segment="seg4"/>
         <l xml:id="frog-L2" xsf:segment="seg5"/>
         <l xsf:segment="seg6"/>
        </q>
       </p>
      </sp>
      <sp who="Louie" xsf:segment="seg7">
       <p xsf:segment="seg7">
        <q xsf:segment="seg7">
         <l xml:id="frog-L1" xsf:segment="seg8"/>
         <l xsf:segment="seg9"/>
        </q>
       </p>
      </sp>
      <sp who="Dewey" xsf:segment="seg10">
       <p xsf:segment="seg10">
        <q xsf:segment="seg10">
         <l xsf:segment="seg11"/>
         <l xml:id="frog-L3" xsf:segment="seg12"/>
        </q>
       </p>
      </sp>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
 </xsf:annotation>
</xsf:corpusData>
```

Listing 13.18: XStandoff-Instanz mit Log

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="sentence">
 <xsf:primaryData start="0" end="24" xml:lang="en" xml:space="preserve" unit="chars">
  <xsf:primaryDataRef uri="sentence.txt" encoding="utf-8" mimeType="text/plain"/>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
  <xsf:segment xml:id="seg2" type="char" start="0" end="3"/>
  <xsf:segment xml:id="seg3" type="char" start="4" end="7"/>
  <xsf:segment xml:id="seg4" type="char" start="8" end="14"/>
  <xsf:segment xml:id="seg5" type="char" start="8" end="13"/>
  <xsf:segment xml:id="seg6" type="char" start="13" end="14"/>
  <xsf:segment xml:id="seg7" type="char" start="15" end="21"/>
  <xsf:segment xml:id="seg8" type="char" start="15" end="20"/>
  <xsf:segment xml:id="seg9" type="char" start="20" end="23"/>
  <xsf:segment xml:id="seg10" type="char" start="21" end="23"/>
 </xsf:segmentation>
 <xsf:annotation>
  <xsf:level xml:id="l_morph">
   <xsf:layer xmlns:morph="http://www.xstandoff.net/morphemes" priority="0">
    <morph:morphemes xsf:segment="seg1">
     <morph:morpheme xsf:segment="seg2"/>
     <morph:morpheme xsf:segment="seg3"/>
     <morph:morpheme xsf:segment="seg5"/>
     <morph:morpheme xsf:segment="seg6"/>
     <morph:morpheme xsf:segment="seg7"/>
     <morph:morpheme xsf:segment="seg10"/>
    </morph:morphemes>
   </xsf:layer>
  </xsf:level>
  <xsf:log>
   <xsf:entry xml:id="ent1" xsf:creator="MStuehrenberg" timestamp="2011-05-19T19:08:00">
    <xsf:add>
     <xsf:level xml:id="l_syll">
      <xsf:layer xmlns:syll="http://www.xstandoff.net/syllables" priority="1">
       <syll:syllables xsf:segment="seg1">
        <syll:syllable xsf:segment="seg2"/>
        <syll:syllable xsf:segment="seg3"/>
        <syll:syllable xsf:segment="seg4"/>
        <syll:syllable xsf:segment="seg8"/>
        <syll:syllable xsf:segment="seg9"/>
       </syll:syllables>
      </xsf:layer>
     </xsf:level>
    </xsf:add>
   </xsf:entry>
  </xsf:log>
 </xsf:annotation>
</xsf:corpusData>
```

13.4 Validierung

Ein wesentliches Merkmal von XSTANDOFF ist die Tatsache, dass es die Validierung von multiplen Annotationen erlaubt. Das bedeutet, dass nicht nur die einzelnen Annotationsebenen auf ihre Gültigkeit hin überprüft werden können, sondern auch die gesamte XSTANDOFF-Instanz (*Cross-Level-* bzw. *Cross-Layer-Validation*, d. h., Validierung über Beschreibungsebenen hinweg).[8] Zu diesem Zweck müssen die ursprünglichen Dokumentgrammatiken der eingebetteten Annotationsebenen angepasst werden (Abschnitt 13.4.2).

13.4.1 XSD als Grammatikformalismus zur Definition von XStandoff

Wie bereits beschrieben, nutzt XSTANDOFF XML SCHEMA als Constraint Language. Die Gründe dafür liegen zum einen an der Nutzung von XML NAMESPACES zur Trennung der einzelnen Level und Layer. Da XML-DTD älter als die entsprechende Spezifikation ist, werden diese nicht unterstützt. Sowohl XSD als auch RELAX NG erlauben dagegen die Verwendung multipler Namensräume. Zum anderen nutzt XSTANDOFF Wildcards, um die entsprechenden Teilbäume zuzulassen. Wie aus dem Abschnitt 3.4 dieser Arbeit ersichtlich, erfüllen ebenfalls sowohl XML SCHEMA als auch RELAX NG diese Vorgabe. Allerdings ermöglicht ersteres eine weiter gehende Steuerung des Validierungsverhaltens mit Hilfe des Attributs `processContents`, das die Werte *strict*, *lax* und *skip* annehmen kann. *strict* verpflichtet den Parser, den Teilbaum anhand des mit dem angegebenen Namensraum assoziierten Schemas zu validieren (im Gegenzug bedeutet das, dass eine solche Dokumentgrammatik in Form eines XSD vorhanden sein muss). Der Wert *lax* impliziert, dass eine Validierung vorgenommen werden soll, sofern ein Schema erreichbar ist (über das Attribut `schemaLocation`) – fehlerhafte Elemente und Attribute werden angemahnt. Fehlt eine Dokumentgrammatik, wird nur auf Wohlgeformtheit hin überprüft, dieses Verhalten wird u. a. beim `meta`-Element des Logs genutzt (vgl. Abschnitt 13.3.1). Im Falle von *skip* wird generell nur eine solche Überprüfung vorgenommen (vgl. die Darstellung des `modify`-Elements in Abschnitt 13.3.4).

RELAX NG kennt eine solche Steuerung nicht, da zum einen die Unterstützung für XML NAMESPACES generell etwas einfacher gehalten ist (wobei diese dennoch vollends unterstützt werden), zum anderen beinhaltet die Spezifikation keine Möglichkeit der Verknüpfung von Schema und Instanz bzw. Namespace und Schema. Eine Verhaltensweise analog zum Wert *lax* lässt sich zwar teilweise emulieren (vgl. Clark und Murata 2001, Abschnitt 11, „Name classes“), für *skip* fehlt aber eine Entsprechung. Da sowohl *lax* als auch *skip* in XSTANDOFF genutzt werden, ist diese Einschränkung problematisch.

Ein weiterer Aspekt von XSTANDOFF, der *für* XML SCHEMA und *gegen* RELAX NG spricht, ist die starke Abhängigkeit von Integritäts-Merkmalen, die unter anderem für die Verbindung von Segment (also Bereich der Primärdaten) und Annotation genutzt

[8] In dieser Arbeit wird ein recht einfaches Konzept von *Cross-Layer-Validation* genutzt. Im Gegensatz dazu verwenden Schonefeld und Witt (2006, S. 2f.) und Tennison (2007) einen engeren Ansatz, der auch gegenseitige Abhängigkeiten von Segmentgrenzen beinhaltet, um ungewollte Überlappungen auszuschließen.

werden. RELAX NG kommt in diesem Aspekt nicht über die Möglichkeiten von XML-DTD hinaus (vgl. die Ausführungen in Abschnitt 3.4).

Von SGF und XStandoff existieren z. Zt. verschiedene Schemata: Da die Entwicklung von SGF als abgeschlossen betrachtet wird, werden die beiden XSD-Dateien (jeweils eine für den *Base Layer* und eine für das Log) nicht mehr weiterentwickelt. XStandoff ist sowohl in einer stabilen, produktiv einsetzbaren Version als auch in einer Entwicklungsfassung verfügbar. Beide sind jeweils einmal mittels XML Schema in der Version 1.0 und in der aktuell noch nicht abgeschlossenen Version 1.1 modelliert (wobei erstere eingebettete Schematron-Assertions enthält, letztere von den in XSD 1.1 eingeführten `assert`-Elementen Gebrauch macht, vgl. Abschnitt 3.4.4.8).[9]

13.4.2 Anpassung der ursprünglichen Dokumentgrammatik

Um die importierten Annotationsebenen als Teilbäume in einer XStandoff-Instanz einbetten und verarbeiten zu können (wozu auch die Validierung zählt), müssen nicht nur die ursprünglichen Instanzen, sondern auch die dazugehörigen Dokumentgrammatiken angepasst werden (vgl. Abschnitt 13.2.3). Auch wenn der Wert des `processContents`-Attribute des Wrapper-Elements `layer` mit dem Wert *lax* belegt ist, wird aufgrund der Verknüpfung von `segment`-Elementen mit den entsprechenden Elementen der Annotationsebenen ein XML-Schema benötigt.[10] Wurde die Dokumentgrammatik mit einer anderen Constraint Language erstellt, so muss sie zunächst in ein XML-Schema umgewandelt werden.[11] Als nächster Schritt erfolgt die eigentliche Anpassung. Dabei wird nach drei Fällen unterschieden:

- Elemente, deren Inhaltsmodell aus anderen Elementen besteht (Container-Elemente), werden durch das optionale `segment`-Attribut aus dem XStandoff-Namensraum erweitert,
- Elemente, deren Inhaltsmodell aus Text besteht (Daten-Elemente), werden zu leeren Elementen umgewandelt und ebenfalls um das optionale `segment`-Attribut erweitert,
- Elemente mit gemischtem Inhaltsmodell werden zu Container-Elementen umgewandelt und ebenfalls um das optionale `segment`-Attribut erweitert.

Das `segment`-Attribut ist Teil einer Attributgruppe `imported`, die zusätzlich noch das optionale Attribut `dcrCategory` zur Referenz einer standardisierten Datenkatego-

[9] Download der XStandoff-Materialien unter `http://www.xstandoff.net/download.html`, zuletzt abgerufen am 19.04.2012.

[10] Natürlich ist es prinzipiell auch möglich, die Integritäts-Beziehung zwischen Segmentierung und Annotationseinheit über einen einfachen Zeichenkettenvergleich durchzuführen. Dazu ist allerdings eine zusätzliche, XML-externe Verarbeitung notwendig, die dem Anspruch von XStandoff, eine vollständig XML-basierte Metasprache zu ermöglichen, zuwiderläuft.

[11] Dies kann z. B. mittels frei verfügbarer Software-Lösungen wie Trang erfolgen. Download und nähere Informationen unter `http://www.thaiopensource.com/relaxng/trang.html`, zuletzt abgerufen am 19.04.2012.

rie enthält.[12] Zu diesem Zweck muss zunächst der Namensraum und das Schema von XStandoff importiert werden (über `xs:import`). Wird der Namensraum zusätzlich noch mit einem Präfix (z. B. `xsf`) eingebunden, kann die das `segment`-Attribut enthaltene Attributgruppe allen Elementen hinzugefügt werden (hierzu müssen zunächst alle Elementdeklarationen zu `xs:complexTypes` umgewandelt werden, da nur solche Attribute beinhalten können).

Da alle Attribute der `imported`-Attributgruppe als optional eingebunden werden, kann das so angepasste Schema sowohl für die ursprüngliche Instanz, als auch für den Teilbaum der XStandoff-Datei zur Validierung herangezogen werden, was eine nachhaltige Nutzung ermöglicht.

13.5 Speicherung

XStandoff-Instanzen können relativ umfangreich werden, woran auch der `all`-Layer nur in Einzelfällen abhelfen kann. Da XStandoff-Dateien sowohl einzelne Korpusdaten als auch theoretisch vollständige Korpora speichern können, ist bereits in der Entwicklung von SGF neben der Speicherung im Dateisystem auch die Verwendung einer nativen XML-Datenbank evaluiert worden (für eine Einführung in die Thematik vgl. Seemann 2003; G. Powell 2007). Einer der Vorteile bei der Verwendung einer nativen XML-Datenbank ist, dass entsprechende Indizes aufgebaut werden, die die Adressierung von Knoten im Baum und damit direkt die Analyse u. a. mit Hilfe des in Abschnitt 14.3 vorgestellten XQuery-Skripts beschleunigen. Umfangreichere Tests wurden mit den nativen Datenbanken eXist[13], Qizx/db 2.1[14] und Oracle Berkeley DB XML[15] absolviert. Darüber hinaus wurden kürzere Evaluationen mit IBM DB2 9.5 Express-C[16] und MarkLogic Server 4.2[17] durchgeführt. Die evaluierten Systeme arbeiteten insgesamt sehr performant, insbesondere in Bezug auf den Arbeitsspeicherverbrauch, der gerade bei Lösungen auf Dateisystem-Ebene oftmals problematisch sein kann (vgl. auch die Ausführungen zu Laufzeit und Korpusgröße in Stührenberg und Goecke 2008). Neuere Entwicklungen, hier vor allem im Bereich der kommenden XSLT-Version (Kay 2010b) erlauben mittels *Streaming XSLT* die Verarbeitung von Instanzen im GB-Bereich ohne den Hauptspeicher übermäßig zu belasten und werden teilweise auch bereits von XSLT-Prozessoren wie Saxon unterstützt (Kay 2010a), so dass zukünftig auch XStandoff-Instanzen mit umfangreichen Annotationen bzw. ganzen Korpora einfacher verarbeitet werden können.[18]

[12] In den letzten beiden Fällen muss streng genommen bis auf das Hinzufügen der `imported`-Attributgruppe keine Änderung vorgenommen werden, da Textknoten auch leer sein können – insofern besteht die Anpassung in allen Fällen im Wesentlichen aus der genannten Erweiterung.

[13] Vgl. `http://www.exist-db.org`, zuletzt abgerufen am 19.04.2012.

[14] Vgl. `http://www.xmlmind.com/qizx/`, zuletzt abgerufen am 19.04.2012.

[15] Vgl. `http://www.oracle.com/us/products/database/berkeley-db/xml/index.html`, zuletzt abgerufen am 19.04.2012.

[16] Vgl. `http://www-01.ibm.com/software/data/db2/`, zuletzt abgerufen am 19.04.2012.

[17] Vgl. `http://www.marklogic.com/products/marklogic-server.html`, zuletzt abgerufen am 19.04.2012.

[18] Für eine entsprechend umfangreich ausgezeichnete XStandoff-Instanz sei auf „Die Bürgschaft“ von Schiller verwiesen, die auf den Ebenen Morpheme, Phrasen, Prosa, Silben, Verse, Worte und

Als Backend für das Web-basierte Annotationswerkzeug SERENGETI wurde dagegen eine Speicherung mittels der relationalen Datenbank MYSQL[19] erfolgreich praktiziert (Poesio *et al.* 2011). Diese Umwandlung ist relativ einfach möglich, da Problemfälle bei der Abbildung von XML-Strukturen auf Tabellen (vgl. Abschnitt 3.1) hauptsächlich bei gemischten Inhaltsmodellen auftreten. Da diese durch die Umwandlung der Inline-Annotation in eine Standoff-Notation in XSTANDOFF nicht existent sind, lassen sich XSTANDOFF-Instanzen bzw. das zu Grunde liegende Datenmodell verhältnismäßig einfach auf die Strukturen einer relationalen Datenbank abbilden. Damit zeigt sich XSTANDOFF insgesamt als äußerst flexibel, was den Aspekt der Speicherung angeht.

XHTML annotiert wurde. Zwar sind die Primärdaten selbst nicht sehr umfangreich, allerdings ist die Menge an durch die Annotation hinzugefügten Informationen um ein Mehrfaches größer. Die ursprünglich in der ersten Phase des „Sekimo"-Projekts entstandenen Annotationsebenen wurde nach XSTANDOFF konvertiert und kann unter `http://www.xstandoff.net/examples/buergschaft.html` eingesehen werden, zuletzt abgerufen am 19.04.2012.

[19] Vgl. `http://mysql.com/`, zuletzt abgerufen am 19.04.2012.

14

Erstellen von und Arbeiten mit XStandoff-Instanzen – das XStandoff-Toolkit

Es dürfte in den bisherigen Abschnitten dieses Teils der Arbeit bereits deutlich geworden sein, dass XStandoff zwar ein durchaus mächtiges Meta-Format darstellt, die Bearbeitung von derart ausgezeichneten Instanzen aber händisch nur unter großem Aufwand möglich ist. Diese Problematik teilt XStandoff mit nahezu allen Standoff-Annotationsformaten (zumindest ist dem Verfasser keine Ausnahme dieser Regel bekannt).

Aus diesem Grunde wurden bereits bei der Entwicklung des Vorläuferformats SGF zunächst rudimentäre Konvertierungsskripte erstellt, die nach und nach erweitert und an XStandoff angepasst wurden und seit Stührenberg und Jettka (2009) als XStandoff-Toolkit (XSFTk) bekannt sind. Das XStandoff-Toolkit besteht zum einen aus den XSLT-2.0-Stylesheets `inline2XSF.xsl` und `mergeXSF.xsl` zur Konvertierung von Inline-Annotationen in das XStandoff-Metaformat. Ebenso enthalten ist das Stylesheet zur Manipulation (Löschen bzw. Extrahieren von Teilbäumen) von XStandoff-Instanzen, `extractXSFcontent.xsl`, sowie mit `XSF2SVG.xsl` ein Skript zur Visualisierung. Die XSLT-Skripte wurden maßgeblich von Daniel Jettka erstellt (zunächst im Rahmen des „Sekimo“-Projekts), der sie für seine Masterarbeit überarbeitet bzw. `XSF2SVG.xsl` neu entwickelt hat (vgl. Jettka 2011). Teilweise wurden vom Verfasser dieser Arbeit Modifikationen vorgenommen.

14.1 Erstellung von XStandoff-Instanzen

Im Folgenden werden die Skripte daher nur kurz vorgestellt, da mit Stührenberg und Jettka (2009) und Jettka (2011) ausführliche Dokumentationen zur Verfügung stehen.

Die Umwandlung einer Inline-Annotation wie z. B. die in den Listings 13.3 und 13.4 gezeigten in eine vollständige XStandoff-Instanz erfolgt in mehreren Schritten:

1. Mittels `inline2XSF.xsl` wird die erste Inline-Annotation in eine XSTANDOFF-Instanz mit einer einzigen Annotationsebene umgewandelt:
   ```
   saxon -o:morph.xsf -s:morph.xml -xsl:inline2XSF.xsl
   primary-data=sentence.txt
   ```
   Dabei ist `saxon` als Aufruf des gleichnamigen XSLT-Prozessors zu verstehen, der über den Paramater `-o` die Ausgabe in die Datei `morph.xsf` schreibt, und als Eingabe die Instanz `morph.xml` nutzt.[1] Das zur Anwendung kommende XSLT-2.0-Stylesheet `inline2XSF.xsl` referenziert über den Parameter `primary-data` die die Primärdaten enthaltene Datei `sentence.txt`. Während der Verarbeitung vergleicht das Skript die Inline-Annotation mit dieser Datei auf Primärdatenidentität, wobei über den zusätzlichen Parameter `pd-check` mittels der Werte *lax*, *middle* oder *strict* die Genauigkeit der Überprüfung variiert werden kann. Ein Wert von *lax* prüft nur auf Vorhandensein aller nicht-Weißraum-Zeichen, der Vorgabewert von *middle* verlangt, dass auch alle Weißraum-Zeichen der Primärdatendatei in der Inline-Annotation enthalten sein müssen, während *strict* zusätzlichen Weißraum nur innerhalb von Elementen mit gemischtem Inhaltsmodell und an den Grenzen von Textknoten erlaubt.

   Nach Abschluss des Vergleichs berechnet das Skript die Segmentgrenzen, erstellt die entsprechenden `segment`-Elemente, passt die ursprüngliche Annotation entsprechend an, und speichert diese unterhalb eines `layer`-Elements in der Ausgabedatei.

2. Der Vorgang wird für die zweite Inline-Annotation wiederholt:
   ```
   saxon -o:syll.xsf -s:syll.xml -xsl:inline2XSF.xsl
   primary-data=sentence.txt
   ```

3. Anschließend werden die beiden XSTANDOFF-Instanzen `morph.xsf` und `syll.xsf`, die jeweils eine einzelne Annotationsebene beinhalten, in eine gemeinsame Datei überführt. Hierzu wird das XSLT-2.0-Stylesheet `mergeXSF.xsl` verwendet:
   ```
   saxon -o:morph-syll.xsf -s:morph.xsf -xsl:mergeXSF.xsl
   merge-with=syll.xsf
   ```
   Mittels `merge-with`-Parameter wird die zweite Datei übergeben. Die Transformation führt zunächst die Segmentlisten zusammen und passt `segment`-Elemente mit gleichem Bezeichner aber unterschiedlichen Start- und Endpositionen entsprechend an. Die gleiche Anpassung wird in der jeweiligen Annotationsebene durchgeführt. Zu beachten ist, dass `mergeXSF.xsl` immer nur zwei Dateien gleichzeitig transformieren kann (die Eingabedatei, im Beispiel `morph.xsf`, und die über `merge-with` angegebene zweite Instanz), so dass der Vorgang mehrfach durchgeführt werden muss, wenn mehr als insgesamt zwei Annotationsebenen kombiniert werden sollen.

Damit ist diese zweistufige Transformation vergleichbar mit dem in Abschnitt 5.6 diskutierten und in den TEI GUIDELINES (P5 1.9.1, Kapitel 16, S. 529f.) definierten Prozess „externalize“.

[1] SAXON hat sich als äußerst performanter XSLT- und XQUERY-Prozessor erwiesen. Darüber hinaus erleichtert die Software mittels eigener Erweiterungen eigene der Transformationsaufgaben.

Das Skript `extractXSFcontent.xsl` dient der Manipulation bestehender XStandoff-Instanzen, indem es das Entfernen von Teilbäumen ermöglicht, die anschließend entweder gelöscht oder in einer eigenen Datei gespeichert werden können. Ein Aufruf wie der nachfolgende würde aus der Datei `morph-syll.xsf` den Teilbaum mit dem Bezeichner `l_morph` entfernen (im Beispiel-Listing 13.10 auf Seite 251 die Morphem-Annotation) und die restliche Instanz als `syll.xsf` speichern:

```
saxon -o:syll.xsf -s:morph-syll.xsf -xsl:extractXSFcontent.xsl
remove-ID=l_morph
```

Wird statt `remove-ID` der Stylesheet-Parameter `extract-ID` verwendet, wird der durch diesen Bezeichner referenzierbare Teilbaum als Ausgabedatei gespeichert (im Beispiel würde die Datei `syll.xsf` also die Morphem-Ebene beinhalten).

Zu Demonstrationszwecken existiert zusätzlich zu den genannten Skripten noch das XSLT-Stylesheet `XSF2inline.xsl`. Es dient der Rücküberführung einer XStandoff-Instanz in eine Inline-Annotation, wobei überlappende Annotationen in Form von TEI-Milestone-Elementen (vgl. Abschnitt 3.3.2.2) repräsentiert werden (ein Beispiel nebst Einführung findet sich in Stührenberg und Jettka 2009). `XSF2Inline.xsl` ist daher vergleichbar mit dem in Abschnitt 5.6 diskutierten und in den TEI Guidelines (P5 1.9.1, Kapitel 16, S. 529f.) definierten Prozess „internalize“.

14.2 Visualisierung von XStandoff-Instanzen

`XSF2SVG.xsl`, das von Daniel Jettka im Rahmen seiner Masterarbeit entwickelt wurde, folgt dem Ansatz aus Piez (2010), in dem verschiedene 2-dimensionale Darstellungen für multiple Annotationen (auf Basis von LMNL, vgl. Abschnitt 3.3.4) diskutiert werden.[2] Aus diesen wurde eine Variante für XStandoff adaptiert und weiterentwickelt. Die Abbildung 14.1 zeigt die Visualisierung für den bekannten Beispielsatz.

Die Darstellung zeigt zunächst die Annotationsebenen (farblich kodiert), mit den dazu gehörigen Elementen. Der Text (die Primärdaten) ist in Zeilen angeordnet, links daneben sind über horizontale Balkendiagramme die jeweiligen Annotationseinheiten dargestellt (zunächst links das Wurzelelement der Ebene, im Beispiel `morphemes` bzw. `syllables` und rechts die entsprechenden Nachkommen). Die Grafik ist interaktiv, d. h., per Mouse-Over-Effekt werden die Elemente und der zugehörige Teil der Primärdaten rot hinterlegt. Darüber hinaus können über die Schalter „display overlaps“ (im Beispiel aktiv) und „display line for charakter position“ Überlappungen grau schattiert hervorgehoben bzw. zusätzliche vertikale Linien für die bessere räumliche Einordnung einzelner Zeichen hinzugefügt werden.

Ausgehend von dieser Arbeit diskutieren Jettka und Stührenberg (2011) weitere Möglichkeiten (auch im dreidimensionalen Raum), um multiple Annotationen graphisch aufbereiten zu können. Eine aktuelle prototypische Implementierung (ebenfalls auf Basis eines XSLT-Stylesheets) ordnet die in einer XStandoff-Instanz gespeicherten Teilbäume der Einzelannotationen hintereinander auf der z-Achse im Raum an. Das Ergebnis der Transformation wird als X3D-Instanz (ISO/IEC 19776-1:2009, eine auf XML

[2] Unter `http://piez.org/wendell/papers/dh2010/clix-sonnets/index.html` sind Beispielinstanzen einsehbar, zuletzt abgerufen am 19.04.2012.

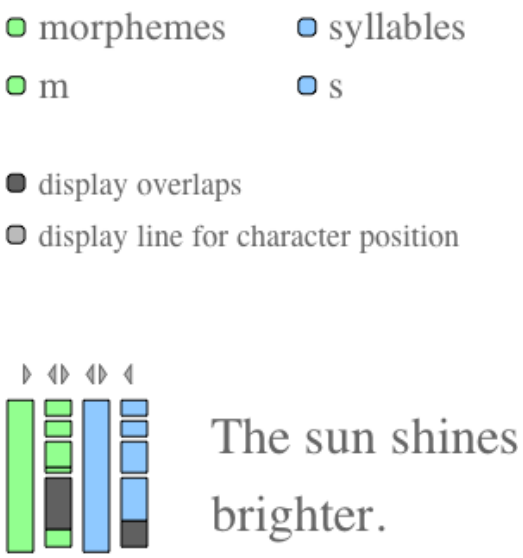


Abbildung 14.1: SVG-Visualisierung der Beispielinstanz aus Listing 13.10

basierende Auszeichnungssprache zur Darstellung drei-dimensionaler Objekte) in eine HTML5-Datei eingebettet. Dank der zunehmenden Unterstützung von HTML5 und der WebGL-Programmierschnittstelle (Marrin 2011), lässt sich die grafische Darstellung in aktuellen Browsern frei in drei Dimensionen drehen.[3] Abbildung 14.2 zeigt die Ausgabe für den Beispielsatz.

Die Blätter des Baumes sind in der Horizontalen oberhalb der Mitte des – nicht visualisierten – Bereichs der Primärdaten angeordnet, so dass nebeneinander sichtbare Knoten aus zwei verschiedenen Annotationsebenen Hinweise auf mögliche Überlappungen geben. Diese können über die Schieberegler am rechten Rand (alternativ über den Menüpunkt „Layers/Merge Layers“) auf der Z-Achse zur Deckungsgleichheit gebracht werden, um überlappende Strukturen deutlicher hervorzuheben. Die Abbildung 14.3 zeigt den entsprechenden Ausschnitt, nachdem Silben- und Morphem-Ebene entsprechend angeordnet wurden.

Wie aus den Abbildungen ersichtlich, werden Mouse-Over-Tooltips verwendet, um nähere Informationen zu den Knoten im Baum anzuzeigen. Dazu gehören der qualifizierte Elementname (d. h., inkl. Namespace-Präfix), der davon umspannte Primärdatenbereich und ein XPATH-Ausdruck zur Bestimmung der Position in der Baumrepräsentation dieser Annotationsebene. Obwohl diese Darstellung aktuell nur zu Demonstrationszwecken entwickelt wurde, zeigt sie zweierlei: Zum einen die grundlegende Eignung von XSTANDOFF als Ausgangspunkt für Visualisierungen von multiplen Hierarchien, zum anderen den Gewinn, den die zusätzliche Achse im Raum für die Darstellung potentiell überlappender Strukturen erbringt.

[3] Getestet mit Google Chrome ab der Version 12 und Firefox aber der Version 4.

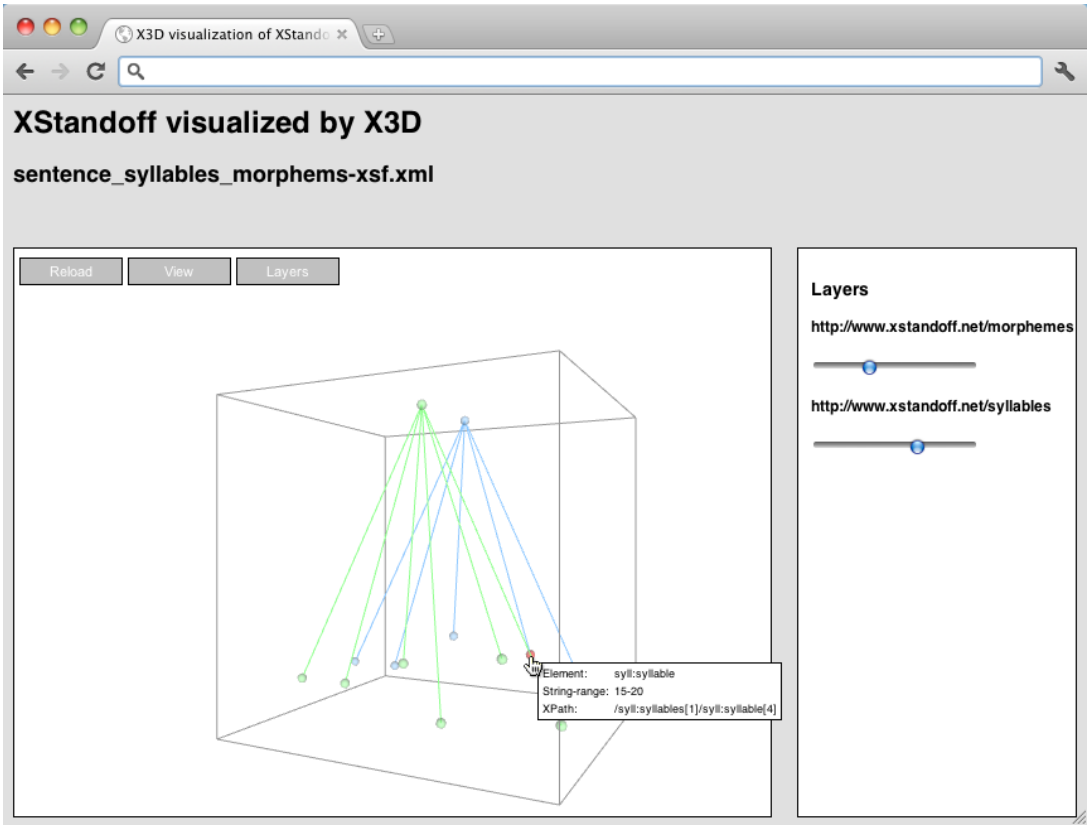


Abbildung 14.2: Prototypische X3D-Visualisierung einer XStandoff-Instanz

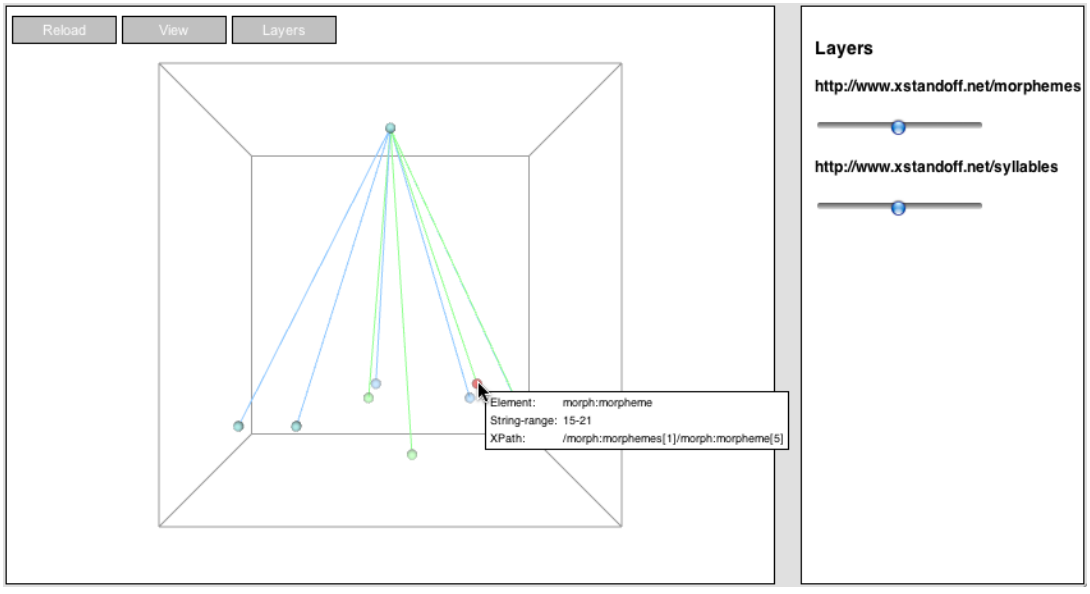


Abbildung 14.3: Deckungsgleiche grafische Darstellung der Morphem- und Silbenebene (bezogen auf die Z-Achse)

14.3 Analyse mittels XQuery

Die Analyse von XSTANDOFF-Instanzen erlaubt das vom Verfasser entwickelte XQUERY-Skript `analyseXSF.xq`. Es verarbeitet eine oder mehrere XSTANDOFF-Dateien und vergleicht ausgehend von einem definierten Element (*targetElement* genannt – standardmäßig das erste Element des ersten Layers, das keine weiteren Kindelemente mehr hat) die Start- und Endpositionen aller Elemente jeglicher Annotationsebenen anhand der jeweiligen Werte der referenzierten Segmente.[4] Die Ergebnisdatei listet anschließend einige der in Abschnitt 3.3.2 aufgeführten Relationen zwischen den Elementen übersichtlich auf. Für die Beispielinstanz aus Listing 13.10 ist die Aufstellung in Listing 14.1 dargestellt.

Listing 14.1: Ergebnis der XQuery-Analyse

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <results>
3   <relations docID="sentence">
4    <targetElement xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" start="0" name="m:m
         " end="3" xsf:segment="seg2">
5     <identical start="0" name="s:s" end="3" xsf:segment="seg2"/>
6     <startPointIdentical start="0" name="m:morphemes" end="24" xsf:segment="seg1"/>
7     <startPointIdentical start="0" name="s:syllables" end="24" xsf:segment="seg1"/>
8    </targetElement>
9    <targetElement xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" start="4" name="m:m
         " end="7" xsf:segment="seg3">
10    <identical start="4" name="s:s" end="7" xsf:segment="seg3"/>
11   </targetElement>
12   <targetElement xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" start="8" name="m:m
         " end="13" xsf:segment="seg5">
13    <startPointIdentical start="8" name="s:s" end="14" xsf:segment="seg4"/>
14   </targetElement>
15   <targetElement xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" start="13" name="
         m:m" end="14" xsf:segment="seg6">
16    <endPointIdentical start="8" name="s:s" end="14" xsf:segment="seg4"/>
17   </targetElement>
18   <targetElement xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" start="15" name="
         m:m" end="21" xsf:segment="seg7">
19    <startPointIdentical start="15" name="s:s" end="20" xsf:segment="seg8"/>
20    <overlap start="20" name="s:s" end="23" xsf:segment="seg9"/>
21   </targetElement>
22   <targetElement xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" start="21" name="
         m:m" end="23" xsf:segment="seg10">
23    <endPointIdentical start="20" name="s:s" end="23" xsf:segment="seg9"/>
24   </targetElement>
25  </relations>
26 </results>
```

Relevant ist hier die Überlappungsrelation (Element `overlap` in Zeile 20) zwischen dem Element `m` der morphologischen Annotationsebene (Position 15 bis 21) und dem Element `s` der Silbenannotation (Position 20 bis 23). Das Skript wurde als Ausgangspunkt für weitere Analysen entwickelt und steht wie die restlichen in diesem Kapitel diskutierten Teile des XSTANDOFF-Toolkits auf der XSTANDOFF-Homepage zum Download zur Verfügung.

[4] Das zu vergleichende Element kann im Skript selbst einfach angepasst werden.

15 XStandoff in der Anwendung

In diesem Kapitel werden exemplarisch zwei weitere Anwendungsbeispiele diskutiert, um die Vielseitigkeit des Formats aufzuzeigen.[1] Zum Einsatz von XStandoff in aktuellen Projekten sei auf den Abschnitt 15.4 verwiesen.

15.1 Alternative Lesarten

Ein verbreitetes linguistisches Phänomen sind alternative Lesarten einer Phrase oder eines Satzes. Der folgende deutsche Satz soll als Beispiel dienen:

> Der Mann sah die Frau mit dem Fernglas.

Das hier zu beobachtende Phänomen ist als PP-Attachment bekannt: Die Präpositionalphrase „mit dem Fernglas“ kann sowohl als Teil der Verbalphrase als auch als Teil der Nominalphrase angesehen werden. Im ersten Fall benutzt das Subjekt (der Mann) ein Fernglas, um die Frau zu sehen – die PP modifiziert also das Verb. Im zweiten Fall erweitert die PP das Akkusativobjekt (die Frau), der Mann sieht also eine Frau, die ein Fernglas (bei sich) trägt. Entsprechende Produktionsregeln einer kontextfreien Grammatik sähen wie folgt aus:[2]

VP → VP, PP
VP → V, NP
NP → Det, N
NP → NP, PP

[1] Weitere Beispiele sind auf `http://www.xstandoff.net/examples/` zu finden, zuletzt abgerufen am 19.04.2012. Die Auflistung in Sperberg-McQueen und Huitfeldt (2012) ermöglicht einen Vergleich von XStandoff und anderen der in Abschnitt 3.3.2 diskutierten Ansätze anhand einfacher Beispiele.

[2] Es wurden im Beispiel bewusst Produktionsregeln für binäre Bäume verwendet, andere Regeln wären ebenso denkbar, z. B. VP → V, NP, PP.

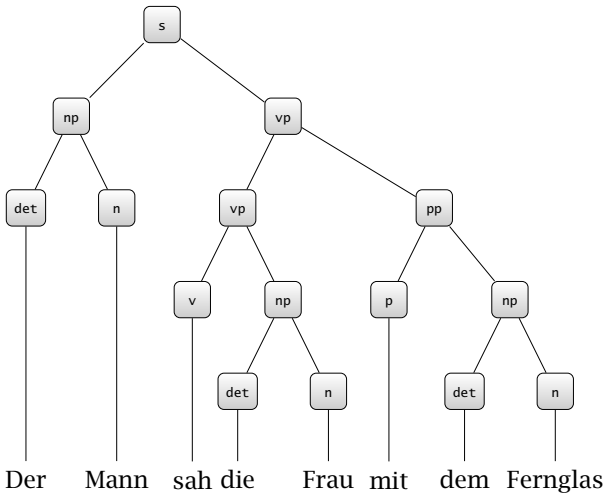


Abbildung 15.1: Baumdarstellung der ersten Lesart

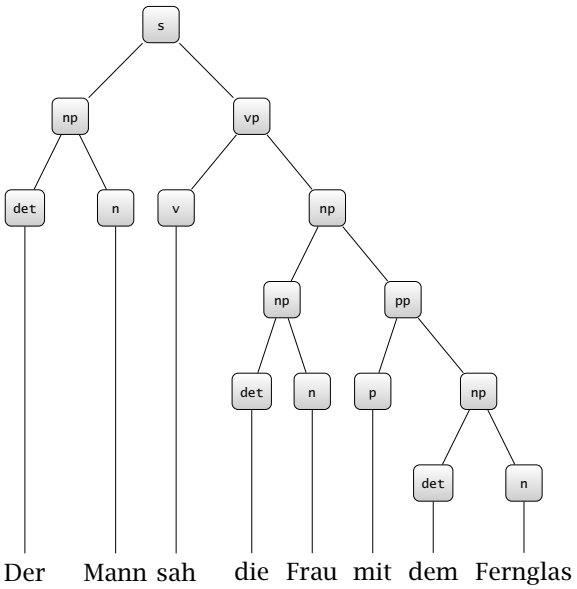


Abbildung 15.2: Baumdarstellung der zweiten Lesart

Die beiden Baumdarstellungen in Abbildung 15.1 und 15.2 zeigen den Unterschied. Diese Ambiguität kann aufgrund von Überlappungen nicht in einer einzelnen Inline-Instanz gespeichert werden. In XSTANDOFF ist es dagegen möglich, Alternativ-Repräsentationen einer Annotationsebene (`level`) unterhalb des gleichen Elements (aber in unterschiedlichen `layer`-Elementen) zu speichern, wie das Listing 15.1 auf Seite 276 zeigt.

Durch die Verwendung von Metadaten (vgl. Abschnitt 13.3.1) können die Varianten weiter beschrieben werden. Einen Schritt weiter geht der experimentelle `all`-Layer, der in XSTANDOFF eingeführt wurde (vgl. Abschnitt 13.3.3) – Listing 15.2 auf Seite 277 zeigt

den angepassten Teil der XStandoff-Instanz.

Unter Berücksichtigung des `all`-Layers erfüllt XStandoff sogar die in Kountz *et al.* (2008) postulierte Voraussetzung, dass unnötige Dopplungen bei der Darstellung alternativer Lesarten vermieden werden sollen (vgl. auch Abschnitt 9.3). Zu beachten ist allerdings, dass bei Verwendung des `all`-Layers eine Validierung (vgl. Abschnitt 13.4) desselben nicht mehr möglich ist, da für diesen zunächst eine Dokumentgrammatik angepasst werden müsste – wobei die Anpassungen deutlich über die in Abschnitt 13.4.2 aufgeführten und ggf. automatisiert durchführbaren Umwandlungen hinausgingen. Gleiches gilt für die anderen Annotationsebenen, deren Elemente teilweise in den `all`-Layer übernommen wurden, da auch hierfür die ursprünglich angepassten XML-Schemata – aufgrund dann evtl. fehlender Kindelemente – nicht mehr adäquat sind. Insofern beschränkt sich die Validierung auf den Base-Layer und evtl. vorhandene Metadaten (sofern für letztere entsprechende Dokumentgrammatiken vorliegen). Darüber hinaus können hierarchische Beziehungen zwischen Elementen verschiedener Layer nur noch aufgrund der Segmentierung (durch Ermittlung von virtuellen Elementbeziehungen, vgl. Abschnitt 14.3) rekonstruiert werden, womit ein Vorteil von XStandoff gegenüber anderen Formaten, die Unterscheidung zwischen Dominanz und Einschluss (vgl. Abschnitt 13.3.2), verloren geht.

Listing 15.1: XStandoff-Instanz mit beiden Lesarten

```
<!-- [...] -->
<xsf:annotation>
 <xsf:level xml:id="linguistic_example_pos-1-level1">
  <xsf:layer xmlns="http://www.xstandoff.net/ambiguity/pos" priority="0">
   <S xsf:segment="seg1">
    <NP xsf:segment="seg2">
     <Det xsf:segment="seg3"/>
     <N xsf:segment="seg4"/>
    </NP>
    <VP xsf:segment="seg5">
     <VP xsf:segment="seg6">
      <V xsf:segment="seg7"/>
      <NP xsf:segment="seg9">
       <Det xsf:segment="seg10"/>
       <N xsf:segment="seg11"/>
      </NP>
     </VP>
     <PP xsf:segment="seg12">
      <P xsf:segment="seg13"/>
      <NP xsf:segment="seg14">
       <Det xsf:segment="seg15"/>
       <N xsf:segment="seg16"/>
      </NP>
     </PP>
    </VP>
   </S>
  </xsf:layer>
 </xsf:level>
 <xsf:level xml:id="linguistic_example_pos-2-level1">
  <xsf:layer xmlns="http://www.xstandoff.net/ambiguity/pos" priority="0">
   <S xsf:segment="seg1">
    <NP xsf:segment="seg2">
     <Det xsf:segment="seg3"/>
     <N xsf:segment="seg4"/>
    </NP>
    <VP xsf:segment="seg5">
     <V xsf:segment="seg7"/>
     <NP xsf:segment="seg8">
      <NP xsf:segment="seg9">
       <Det xsf:segment="seg10"/>
       <N xsf:segment="seg11"/>
      </NP>
      <PP xsf:segment="seg12">
       <P xsf:segment="seg13"/>
       <NP xsf:segment="seg14">
        <Det xsf:segment="seg15"/>
        <N xsf:segment="seg16"/>
       </NP>
      </PP>
     </NP>
    </VP>
   </S>
  </xsf:layer>
 </xsf:level>
</xsf:annotation>
<!-- [...] -->
```

Listing 15.2: XStandoff-Instanz mit beiden Lesarten und `all`-Layer

```
<!-- [...] -->
<xsf:annotation>
 <xsf:level xml:id="linguistic_example_pos-1-level">
  <xsf:layer xmlns:all="http://www.xstandoff.net/2009/all" priority="0">
   <all:S xsf:segment="seg1">
    <all:NP xsf:segment="seg2">
     <all:Det xsf:segment="seg3"/>
     <all:N xsf:segment="seg4"/>
    </all:NP>
    <all:VP xsf:segment="seg5"/>
    <all:PP xsf:segment="seg12">
     <all:P xsf:segment="seg13"/>
     <all:NP xsf:segment="seg14">
      <all:Det xsf:segment="seg15"/>
      <all:N xsf:segment="seg16"/>
     </all:NP>
    </all:PP>
   </all:S>
  </xsf:layer>
  <xsf:layer xmlns="http://www.xstandoff.net/linguistic_example/pos" priority="0">
   <VP xsf:segment="seg6">
    <V xsf:segment="seg7"/>
    <NP xsf:segment="seg9">
     <Det xsf:segment="seg10"/>
     <N xsf:segment="seg11"/>
    </NP>
   </VP>
  </xsf:layer>
  <xsf:layer xmlns="http://www.xstandoff.net/linguistic_example/pos" priority="0">
   <V xsf:segment="seg7"/>
   <NP xsf:segment="seg8">
    <NP xsf:segment="seg9">
     <Det xsf:segment="seg10"/>
     <N xsf:segment="seg11"/>
    </NP>
   </NP>
  </xsf:layer>
 </xsf:level>
</xsf:annotation>
<!-- [...] -->
```

15.2 Multimodale Annotationen

Multimodale Daten stellen besondere Anforderungen an die Erstellung eines Korpus, da üblicherweise mehrere Primärdaten verwendet werden. So kommen in multimodalen Annotationen sowohl Audio- und/oder Videodateien als auch davon abgeleitete Transkriptionen (üblicherweise in Form von XML-Annotationen) zum Einsatz.

In einer Beispiel-Instanz werden hierzu zwei Eigenschaften von XSTANDOFF demonstriert: (1) die Möglichkeit, mit Hilfe des Elements `corpus` mehrere Korpuseinträge (sprich: mehrere `corpsuData`-Elemente) zusammenfassen zu können, und (2) die Verwendung mehrerer Primärdaten pro Korpuseintrag. Die Daten (sowohl Primärdaten als auch Annotation) hierzu wurden aus dem B1-Projekt („Speech-gesture alignment“) des Sonderforschungsbereich 673 „Alignment in Communication“ dankenswerterweise zur Verfügung gestellt.[3] Listing 15.3 zeigt eine gekürzte Version.

Listing 15.3: Multimodale Annotation in XStandoff

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpus xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1">
 <xsf:corpusData xml:id="mm-timeline" xsfVersion="1.1">
  <xsf:primaryData start="0" end="335587" unit="milliseconds" xml:id="pd1_video" role="
      master">
   <xsf:primaryDataRef uri="b1-video.mpg" mimeType="video/mpeg"/>
  </xsf:primaryData>
  <xsf:primaryData start="0" end="335587" unit="milliseconds" xml:id="pd1_video_a" role="
      slave">
   <xsf:primaryDataRef uri="b1-video_a.mpg" mimeType="video/mpeg"/>
  </xsf:primaryData>
  <xsf:primaryData start="0" end="335587" unit="milliseconds" xml:id="pd1_video_b" role="
      slave">
   <xsf:primaryDataRef uri="b1-video_a.mpg" mimeType="video/mpeg"/>
  </xsf:primaryData>
  <xsf:segmentation>
   <xsf:segment xml:id="pd1_seg1" start="309924" end="310079"/>
   <xsf:segment xml:id="pd1_seg2" start="310079" end="310302"/>
   <!-- [...] -->
   <xsf:segment xml:id="pd1_seg77" start="335307" end="335587"/>
  </xsf:segmentation>
  <xsf:annotation>
   <xsf:level xml:id="gesture">
    <xsf:layer>
     <gestureSequences xmlns="http://www.xstandoff.net/gesture">
      <gestureSequence who="router" xsf:segment="pd1_seg12">
       <leftHandPhrase xsf:segment="pd1_seg12" value="iconic">
        <phase xsf:segment="pd1_seg13" value="prep"/>
        <!-- [...] -->
        <perspective xsf:segment="pd1_seg12" value="speaker"/>
       </leftHandPhrase>
       <rightHandPhrase xsf:segment="pd1_seg12" value="iconic">
        <phase xsf:segment="pd1_seg13" value="prep"/>
        <!-- [...] -->
        <perspective xsf:segment="pd1_seg12" value="speaker"/>
       </rightHandPhrase>
      </gestureSequence>
      <gestureSequence who="router" xsf:segment="pd1_seg47">
       <leftHandPhrase xsf:segment="pd1_seg48" value="iconic">
        <phase xsf:segment="pd1_seg49" value="prep"/>
```

[3] Nähere Informationen zum Projekt unter `http://www.sfb673.org/projects/B1/`, zuletzt abgerufen am 19.04.2012.

```
38        <!-- [...] -->
39        <practice xsf:segment="pd1_seg48" value="grasping"/>
40       </leftHandPhrase>
41       <rightHandPhrase xsf:segment="pd1_seg48" value="iconic">
42        <phase xsf:segment="pd1_seg49" value="prep"/>
43        <!-- [...] -->
44        <practice xsf:segment="pd1_seg48" value="grasping"/>
45       </rightHandPhrase>
46       <!-- [...] -->
47      </gestureSequence>
48      <!-- [...] -->
49     </gestureSequences>
50    </xsf:layer>
51   </xsf:level>
52   <xsf:level xml:id="transcription">
53    <xsf:layer xmlns:all="http://www.xstandoff.net/2009/all">
54     <text xmlns="http://www.tei-c.org/ns/1.0">
55      <body>
56       <div>
57        <u who="router" xsf:segment="pd1_seg1">ich</u>
58        <u who="router" xsf:segment="pd1_seg2">weiß</u>
59        <!-- [...] -->
60        <u who="router" xsf:segment="pd1_seg77">ja</u>
61       </div>
62      </body>
63     </text>
64    </xsf:layer>
65   </xsf:level>
66  </xsf:annotation>
67 </xsf:corpusData>
68 <xsf:corpusData xml:id="mm-character" xsfVersion="1.1" xsf:alt="multimodal-timeline">
69  <xsf:primaryData start="0" end="303" unit="chars" xml:id="b1_text">
70   <xsf:primaryDataRef uri="b1.txt" mimeType="text/plain" encoding="utf-8"/>
71  </xsf:primaryData>
72  <xsf:segmentation>
73   <xsf:segment xml:id="seg1" start="0" end="302"/>
74   <!-- [...] -->
75   <xsf:segment xml:id="seg56" start="300" end="302"/>
76  </xsf:segmentation>
77  <xsf:annotation>
78   <xsf:level xml:id="b1_transkription-level1">
79    <xsf:layer xmlns="http://www.xstandoff.net/b1/token">
80     <text xsf:segment="seg1">
81      <token type="PPER" lemma="ich" xsf:segment="seg2"/>
82      <token type="VVFIN" lemma="wissen" xsf:segment="seg3"/>
83      <!-- [...] -->
84      <token type="ADV" lemma="ja" xsf:segment="seg56"/>
85     </text>
86    </xsf:layer>
87   </xsf:level>
88  </xsf:annotation>
89 </xsf:corpusData>
90 </xsf:corpus>
```

In diesem Fall gibt es insgesamt zwei Korpuseinträge: Den mit dem Bezeichner *mm-timeline* gekennzeichneten (Zeile 3) und den als *mm-character* bezeichneten in Zeile 68. Letzterer bezieht sich mittels des optionalen Attributs `alt` (in der Dokumentgrammatik als vom Datentyp `xs:IDREFS` deklariert) auf den ersten. Das erste Korpusdatum verweist auf drei Video-Primärdaten, die die Dialogszene in unterschiedlichen Kameraperspektiven aufgezeichnet haben, und von denen eine als „Master“ fungiert (gekennzeichnet durch den Wert des `role`-Attributs), d. h. die Zeitleiste vor-

gibt, anhand derer die Annotationen ausgerichtet werden. Die erste Annotationsebene (beginnend in Zeile 21) beschreibt einzelne Gesten, die vom sogenannten *Router* bzw. dessen *Follower* ausgeführt werden.[4] Diese lassen sich nach linker und rechter Hand unterteilen und weiter untergliedern. Daran schließt sich in Zeile 53 die Annotationsebene der Transkription an, die nach den TEI GUIDELINES ausgezeichnet ist und sich sich nach den Zeitpunkten in der Videoaufnahme richtet. Dagegen nutzt der zweite Korpuseintrag eine separate Transkription (auf Basis einer Textdatei), um diese mittels eines Taggers automatisiert mit syntaktischen Informationen anzureichern (Zeile 79ff.).

Wenngleich auch XSTANDOFF nicht im Projekt B1 zum Einsatz kommt (das Listing 15.3 ist zu rein demonstrativen Zwecken erstellt worden), zeigt das Beispiel doch, dass XSTANDOFF nicht nur für die Annotation und Speicherung textueller Korpora geeignet ist.

15.3 Merkmalsstrukturen

Da XSTANDOFF prinzipiell keinerlei Einschränkungen bezüglich der Teilbäume unterhalb des `layer`-Elements vorsieht, können auch standardisierte Merkmalsstrukturbeschreibungen (vgl. Kapitel 6) genutzt werden. Allerdings erscheint eine solche Kombination dem Verfasser nicht zwingend sinnvoller als die Kombination von XSTANDOFF mit entsprechend den in Abschnitt 13.2.3 genannten angepassten Inline-Annotationen. Im Gegenteil ist zu befürchten, dass durch den Einsatz von ISO-Merkmalsstrukturen die Dateigröße weiter erheblich ansteigen wird, was zu einer verminderten Übersichtlichkeit und Wartbarkeit der resultierenden Instanz führen kann (vgl. auch die Ausführungen in Abschnitt 6.3).

15.4 Verbreitung und Einsatz

Wie bereits angesprochen, wurde XSTANDOFF bzw. dessen Vorgängerformat SGF im Rahmen des „Sekimo“-Projekts entwickelt. Hier kam es zur Speicherung von mehrfach annotierten linguistischen Korpora und deren anschließender Analyse in der linguistischen Domäne der Anaphernauflösung zum Einsatz (vgl. Stührenberg und Goecke 2008). Dabei wurden für die jeweiligen Korpuseinträge zunächst normalisierte Primärdaten (in Bezug auf Weißraum-Zeichen) in Form von Textdateien erstellt, die dann mittels verschiedener linguistischer Ressourcen (bzw. im Fall einer Ebene auch manuell) annotiert wurden. Die resultierenden Annotationsebenen (logische Dokumentstruktur, Depenz-Parsebaum, Diskursentitäten und Annotation anaphorischer Relationen) wurden mittels `inline2XSF.xsl` und `mergeXSF.xsl` (bzw. mit Hilfe der zum damaligen Zeitpunkt für SGF äquivalenten Varianten) in SGF-Instanzen überführt (für eine ausführlichere Darstellung des Korpus vgl. Witt, Goecke, Stührenberg *et al.* 2011). Diese wurden dann auf zweierlei Weise eingesetzt: Zum einen dienten sie (ohne die Auszeichnungsebene der anaphorischen Relationen) als Import- und Exportformat für das bereits erwähnte Annotationswerkzeug SERENGETI, mit dem sie dann um die

[4] Im Projekt wird der prototypische Dialog einer Wegbeschreibung analysiert.

Anaphern-Ebene ergänzt wurden. Zum anderen, um Beziehungen zwischen Elementen unterschiedlicher Annotationsebenen zu identifizieren, beispielsweise Auswirkungen der logischen Dokumentstruktur auf die Kandidatensuche für anaphorische Relationen. Zu diesem Zweck wurden Vorläufer des in Abschnitt 14.3 demonstrierten XQuery-Skripts entwickelt (teilweise noch auf Basis von XSLT), die Antezedens-Kandidaten für Anaphern einer gegebenen anaphorischen Relation in Abhängigkeit ihrer Position in der logischen Dokumentstruktur ermitteln (wie z. B. Position im Satz, Absatz, etc., eine Beispielkandidatenliste ist enthalten in Witt, Goecke, Stührenberg *et al.* 2011).

Nach Abschluss des Projekts wurde der „Sekimo"-Korpus (zu diesem Zeitpunkt 14 Instanzen, darunter sechs deutschsprachige wissenschaftliche Artikel und acht deutsche Zeitungsartikel – insgesamt 3084 Sätze, 55221 Token, 4185 anaphorische Relationen) nach XStandoff überführt und um weitere kleinere Korpuseinträge (deutsch und englisch) erweitert.[5] Aufgrund einer Kooperation mit den Entwicklern der „AnaphoricBank" (vgl. Poesio *et al.* 2011) wurden ebenfalls einige Korpuseinträge des XStandoff-Korpus hinzugefügt. XStandoff war darüber hinaus als eines von mehreren Exportformaten für die „AnaphoricBank" im Gespräch, der aktuelle Stand ist allerdings ungeklärt.

Lüngen und Lobin (2011) erwähnen die Anwendung von XStandoff im Rahmen der Extraktion von Domänenwissen aus Inhaltsverzeichnissen. Auch hier sind die Ausgangsdaten mit verschiedenen Annotationsebenen versehen, die mittels XStandoff verarbeitet werden (vgl. auch Lüngen und Hebborn 2010).

Davon unabhängig wurde seitens des Software-Entwicklers Gregor Middell eine Java-Klasse entwickelt, um LMNL-Instanzen in Form eines *LMNL Object Model* (LOM) (zu LMLNL vgl. Abschnitt 3.3.2) nach XStandoff zu überführen.[6] Nähere Hintergründe zur Motivation sind dem Verfasser allerdings nicht bekannt. Da XStandoff vollständig unter der *GNU Lesser General Public License* (LGPL v3) offen zum Download zur Verfügung steht, ist ein Einsatz in weiteren Projekten ohne Wissen des Verfassers denkbar (und rechtlich möglich).

15.5 Beurteilung

Im Abschnitt 3.3 wurden mögliche formale Modelle für XML-Instanzen diskutiert, mit dem Ergebnis, dass eine Erweiterung gegenüber des Datenmodell des Baums (in Form des in Abschnitt 3.3.3 als *eXtended Tree*, XT, bezeichneten allgemeineren gerichteten Graphen) abhängig von der Verwendung eines Grammatikformalismus' ist, der die Überprüfung von Integrität unterstützt (z. B. durch entsprechende Datentypen). Auch XStandoff macht davon Gebrauch, indem die Beziehung zwischen Segmenten und Auszeichnung durch eben diesen Mechanismus hergestellt wird. Zu unterscheiden sind hier zwei Aspekte: Eine XStandoff-Instanz ist technisch gesehen ein erweiterter Baum, also ein geordneter Baum mit zusätzlichen Link-Kanten. Allerdings unterstützt das Metaformat sowohl Self-Overlap (vgl. Abschnitt 3.3.2) als auch diskontinuierliche

[5] Aus rechtlichen Gründen ist nur eine Auswahl an Beispielen ist auf der XStandoff-Webseite unter `http://www.xstandoff.net/examples/index.html` einsehbar, zuletzt abgerufen am 19.04.2012.

[6] Details der Java-Dokumentation sind unter `http://www.middell.net/lmnl4j/apidocs/org/lmnl/xml/xsf/package-summary.html` zu finden, zuletzt abgerufen am 19.04.2012.

Annotationseinheiten. Damit geht es über die Möglichkeiten eines Clean r-GODDAGs und der allgemeineren Form des r-GODDAG hinaus, da letzterer das Vorliegen diskontinuierlicher Einheiten ausschließt (vgl. Abschnitt 3.3.3.1). Somit bleibt bei den vorgestellten Modellen das des Generalized GODDAGs als nahe liegende Entsprechung.

Wie in Abschnitt 13.4 dargelegt, liegt die XStandoff definierende Dokumentgrammatik in Form mehrerer XML-Schemata vor; die aktuelle stabile Version ist mittels XML Schema in der Version 1.0 formalisiert, eine Entwicklerversion nutzt die kommende Version 1.1. Erstere kann formal als Vertreter einer Restrained-Competition Tree Grammar (RTG) eingeordnet werden, da die Elemente `meta` und `layer` von Element-Wildcards Gebrauch machen. Darüber hinaus werden eingebettete Schematron-Regeln genutzt, um z. B. sicherzustellen, dass bei mehreren `primaryData`-Elementen nur eines als „Master“ ausgezeichnet ist (vgl. das Anwendungsbeispiel in Listing 15.3). Weitere Regeln sichern zu, dass keine Segmente existieren, deren Start- bzw. Endposition kleiner bzw. größer als die der Primärdaten ist. Die Entwicklerversion setzt für diese Co-occurrence Constraints (vgl. Abschnitt 3.4.4.4) die mit XSD 1.1 eingeführten `assert`-Elemente ein und unterscheidet sich sonst nur in Nuancen von der stabilen Version. Die hierzu modellierte Dokumentgrammatik kann von der formalen Ausdrucksstärke her als die in Abschnitt 3.4.4.8 eingeführte Unambigious Restrained Competition Grammar (URCG) eingestuft werden. Damit nutzt XStandoff die Ausdrucksstärke von XML Schema effizient aus.

Die Möglichkeiten zur Anknüpfung an eine standardisierte Datenkategorieauswahl (DCS) nach ISO 12620:2009 (vgl. Kapitel 8 und Abschnitt 13.2.3), ermöglichen nicht nur die Einbettung einer XStandoff-Instanz in das Normen-Ökosystem, sondern können auch als Basis für einen Vergleich von lokalen und öffentlichen Kategorien dienen. Die darüber hinaus stringente Trennung von Level und Layer, also der in Abschnitt 3.5 diskutierten Unterscheidung zwischen konzeptueller Ebene und deren Serialisierung(en) ermöglicht gewisse Freiheiten und erlaubt auf relativ einfache Weise den Vergleich von Annotationen (sowohl automatisiert oder manuell), wie in Abschnitt 14.3 gezeigt wurde. Dazu kann auch die Visualisierung von XStandoff-Instanzen beitragen.

Abschließend beurteilt nutzt XStandoff sowohl in der Instanz als auch auf Seiten der Constraint Language mächtige und ausdrucksstarke Formalismen, schränkt aber aufgrund der fehlenden Vorgabe eines Tagsets den Benutzer bei der Wahl seiner Annotationseinheiten nicht ein und ermöglicht somit die Kombination mit anderen in dieser Arbeit untersuchten Spezifikationen zur Speicherung von linguistischen Korpusdaten. Diese Freiheit bürdet auf der anderen Seite dem Nutzer allerdings auch auf, die Benennung der Einheiten der Informationsmodellierung sorgfältig zu wählen (oder Gebrauch vom für Annotationsebenen global zur Verfügung stehenden `dcrCategory`-Attribut zu machen). Davon abgesehen hat sich XStandoff in mehrjährigem Einsatz bewährt und wird vom Verfasser auch in Zukunft weiter gepflegt werden.

Teil IV

Diskussion und Fazit

16 Diskussion

Wie die bisherigen Kapitel dieser Arbeit demonstriert haben, sind für die linguistische Annotation Spezifikationen und Normen von Relevanz, die in verschiedenen Organisationen verabschiedet wurden und werden. Die drei hier zu nennenden Akteure sind das *Unicode Consortium*, das *World Wide Web Consortium* und die *International Organization for Standardization*. Die durch das *Unicode Consortium* entwickelte gleichnamige Spezifikation (Unicode 1.0; Unicode 6.0.0; ISO/IEC 10646:2011) ermöglicht es, internationale Texte transparent kodieren zu können und bildet neben der durch die ISO entwickelten Metasprache SGML eine der Grundlagen der vom *W3C* verabschiedeten Metasprache XML. Begleitend dazu wurde im Laufe der letzten Dekade hauptsächlich durch das *World Wide Web Consortium* eine ganze Familie von ergänzenden Spezifikationen verabschiedet, darunter Standards zum Zwecke der Validierung (XML Schema), der Navigation (XPath), der Transformation (XSLT) und der gezielten Abfrage (XQuery) – grundlegende Arbeiten, die die Entwicklung der darauf aufbauenden und in den Teilen II und III dieser Arbeit vorgestellten Spezifikationen erst möglich gemacht und deren Verbreitung und Nutzen erheblich gesteigert haben. Darüber hinaus beeinflussen durch das W3C erarbeitete Spezifikationen wie das Internationalization Tag Set (ITS, Lieske und Sasaki 2007) aber auch weniger generische Aspekte der Arbeit mit linguistischen Daten.

Ähnlich verläuft die Normierungsarbeit innerhalb der *International Organization for Standardization*. Auch hier finden sich grundlegende Standards wie die Constraint Languages RELAX NG und Schematron, die Gegenstand der Normierung im JTC 1/SC 34 sind, dem gleichen Gremium, das bereits SGML oder auch DSSSL verabschiedet hat. Alle diese grundlegenden Spezifikationen wurden bereits im Kapitel 3 diskutiert. Das Hauptaugenmerk dieser Arbeit liegt allerdings auf den im Rahmen von TC 37 verabschiedeten Normen, die aufbauend auf den genannten Pfeilern spezifische Aspekte linguistischer Auszeichnung betreffen und vorrangig im Unterkomitee SC 4 erarbeitet (mit Ausnahme von ISO 12620:2009, das in SC 3 normiert wurde) und im Teil II vorgestellt wurden. Allerdings lassen sich diese Standards nicht losgelöst betrachten, vielmehr haben die

ebenfalls diskutierten TEI GUIDELINES, sowie der CORPUS ENCODING STANDARD bzw. dessen Übearbeitung XCES wichtige Einflüsse auf die Normungsarbeiten gehabt. Im Folgenden sollen daher zum einen diese Zusammenhänge aber auch die Beziehungen zwischen den in dieser Arbeit diskutierten Normen zusammengefasst werden, um einen abschließenden Überblick über die doch sehr komplexe Landschaft der Standards für linguistische Ressourcen zu ermöglichen.

16.1 Einflüsse der De-facto-Standards auf die Arbeiten in TC 37/SC 4

Abgesehen von der eindeutigen Beziehung zwischen der Version P3 der TEI GUIDELINES und dem CORPUS ENCODING STANDARD als deren Modifikation, wurden auch die Arbeiten in TC 37/SC 4 maßgeblich durch die *Text Encoding Initative* beeinflusst. Diese Entwicklungen (nicht immer kann man von eindeutig zuzuordnenden Einflüssen sprechen) lassen sich anhand der in Kapitel 3 diskutierten vier Komponenten von Auszeichnungssprachen (die Syntax bzw. Notation, das formale Modell, die Dokumentgrammatik und die Trennung von Konzept und Serialisierung) festmachen.

16.1.1 Syntax und Notation

Die Syntax wird bestimmt von der verwendeten Metasprache (vgl. Abschnitt 3.2) und kann daher für die Betrachtung der in dieser Arbeit thematisierten Spezifikationen relativ kurz abgehandelt werden. So sind die De-facto-Standards TEI GUIDELINES in den Versionen P1 bis P4-Beta (vgl. Kapitel 5.1) in Form von SGML-Tagsets modelliert. Als Modifikation der P3 verwendet CES die gleiche Metasprache. Dagegen sind sowohl XCES als auch die 2002 veröffentlichte Fassung P4 (und nachfolgende Versionen) der TEI GUIDELINES XML-Anwendungen – wie alle anschließend entwickelten Normen und Spezifikationen, die in dieser Arbeit diskutiert wurden.

Relevanter sind die Veränderungen, die im Rahmen der Notation zu beobachten sind. Die TEI GUIDELINES – als Auszeichnungssprache klassischer Prägung – verwenden eine Inline-Notation, d. h., die Markierungen werden in den textuellen Primärdaten in Form von Start- und Endtags um die auszuzeichnenden Inhalte herum angeordnet. Da die Auszeichnung linguistischer Daten in Inline-Notation verstärkt in überlappenden Strukturen resultieren kann (Sasaki 2004, S. 209), führen bereits die TEI GUIDELINES entsprechende Maßnahmen zur Vermeidung eben solcher an (vgl. Abschnitt 3.3.2). Interessant ist, dass von den genannten Konzepten (multiple Dokumente, Milestones, Fragmentierung, Standoff-Annotation) drei dazu dienen, die grundlegende Inline-Annotation beibehalten zu können (ähnlich wie einige der zusätzlich in der P5 1.9.1, Abschnitt 20.5, „Non-XML-based Approaches“ oder in Stührenberg und Goecke (2008) aufgeführten nicht auf XML-basierenden Auszeichnungssprachen, die aber nicht im Fokus dieser Arbeit liegen). Die in Abschnitt 3.3.2.3 diskutierte Standoff-Annotation stellt insofern einen Paradigmenwechsel dar, als durch die Trennung von Annotation und Primärdaten nicht nur die multiple Auszeichnung, sondern prinzipiell auch die Unterscheidung von Konzept und Serialisierung ermöglicht werden (vgl. Abschnitt 16.1.4). CES als P3-Anwendung und dessen Nachfolger XCES propagieren erstmalig die Notation im

Standoff-Verfahren zur konkreten Auszeichnung linguistischer Daten (vgl. Kapitel 7; allerdings ist auch weiterhin ein Inline-Annotationsinventar vorhanden, beispielsweise bei der Strukturierung der Primärdaten). Dieser Ansatz beeinflusst auch die weitere Entwicklung der TEI GUIDELINES: In der aktuellen P5 wird nicht nur der Terminus „Standoff“ als solcher genannt, das Konzept wird vielmehr in mehreren Unterkapiteln und Abschnitten thematisiert (vorrangig in den Kapiteln 16 und 20) und ist auch Gegenstand verschiedener wissenschaftlicher Veröffentlichungen (wie beispielsweise Dipper, Götze, Küssner *et al.* 2007; Bański 2010; Ide, Suderman und Simms 2010, um nur drei zu nennen, vgl. auch die Ausführungen in Abschnitt 5.6).[1] Auch die in Teil II diskutierten und in TC 37/SC 4 erarbeiteten Normen wie das LINGUISTIC ANNOTATION FRAMEWORK, das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK oder das SYNTACTIC ANNOTATION FRAMEWORK setzen auf Standoff-Notation, so dass sich insgesamt die Etablierung dieses Konzepts feststellen lässt, auch Dank leistungsfähigerer Werkzeuge, wie beispielsweise das im „Sekimo“-Projekt erstellte Web-basierte Annotationswerkzeug SERENGETI oder die GLOZZ ANNOTATION PLATFORM[2] (Widlöcher und Mathet 2009), die im Projekt „ANNODIS“ (Annotation Discursive) entwickelt wurde.[3] So bewerten auch Bański und Przepiórkowski (2010a, S. 98) die Unterstützung von Standoff-Annotation als wesentliches Merkmal von Formaten zur Auszeichnung linguistischer Daten: „Any standards adopted for these levels should allow for stand-off annotation, as is now common practice and as is virtually indispensable in the case of many levels of annotation, possibly involving conflicting hierarchies.“

16.1.2 Grammatik

Die Dokumentgrammatik einer Auszeichnungssprache kann in mehrfacher Hinsicht evaluiert werden: Zum einen anhand der eingesetzten Constraint Language, die Rückschlüsse auf die formale Ausdrucksstärke zulässt – wobei die theoretisch mögliche Ausdrucksstärke nicht zwingend mit der tatsächlichen eines konkreten Schemas übereinstimmen muss. Zum anderen deklariert die Grammatik die Symbole in Auszeichnungssprachen, also die Elemente (und deren hierarchischen Beziehungen zueinander) und Attribute. Im Folgenden soll zunächst der erste Aspekt thematisiert werden.

Rein historisch bedingt sind frühere Versionen der TEI GUIDELINES, sowie die ursprüngliche CES-Fassung auf die DTD (zunächst SGML-, später XML-) als Grammatikformalismus beschränkt, da andere Formalismen gar nicht oder nur im Sinne von Forschungsprojekten zur Verfügung standen. Wie in Abschnitt 3.4 dargelegt, entsprechen die zu Grunde liegenden Schemata formal gesehen damit lokalen Baumgrammatiken (LTG). Anders dagegen die aktuelle Fassung P5, die durch ein RELAX-NG-Schema definiert wird. Diese Constraint Language ist prinzipiell mächtig genug, um reguläre Baumgrammatiken (RTG) zu modellieren. Die die P5 definierende Dokumentgrammatik

[1] Da die P4 nur eine Reformulierung der P3 darstellt, ist es nicht verwunderlich, dass der Einfluss des Standoff-Konzepts erst in der P5 deutlich wird.

[2] Der Download der Software sowie weiterer Dokumente ist unter der URL `http://www.glozz.org/` möglich, zuletzt abgerufen am 19.04.2012.

[3] Weitere Informationen zum Projekt unter der URL `http://w3.erss.univ-tlse2.fr/annodis`. Zuletzt abgerufen am 19.04.2012.

wird allerdings automatisch generiert und entspricht von ihrer formalen Ausdrucksstärke her ebenfalls einer LTG (vgl. die Ausführungen in Abschnitt 5.9).

XCES ist mittels XML SCHEMA definiert, ein Formalismus, der die Definition von Grammatiken der formalen Mächtigkeit einer Single Type Tree Grammar (oder darüber hinaus) erlaubt, und macht von Datentypen und einigen weiteren mit diesem Grammatikformalismus eingeführten Merkmalen Gebrauch. Unter formalen Gesichtspunkten betrachtet entspricht die Grammatik allerdings ebenfalls einer lokalen Baumgrammatik, so dass nur technische Gründe für die Verwendung von XSD sprechen (was aber vollends legitim ist; vgl. Abschnitt 7.9).

Die in dieser Arbeit diskutierten Normen definieren – mit Ausnahme des noch nicht final veröffentlichten MORPHO-SYNTACTIC ANNOTATION FRAMEWORK – keine eigene Auszeichnungssprachen.[4] Die MAF-Dokumentgrammatik ist in Form eines RELAX-NG-Schemas modelliert, besitzt aber ebenfalls nur die formale Ausdrucksstärke einer lokalen Baumgrammatik (vgl. Abschnitt 10.3). Hier dürften die Gründe für die Verwendung im Status von RELAX NG als Internationaler Standard liegen. Darüber hinaus ist festzuhalten, dass Dokumentgrammatiken als Teil von Normen oftmals nur Eingang in den informativen Anhang finden, also nicht als normativ anzusehen sind (im Gegensatz zu De-facto-Standards wie den TEI GUIDELINES, die konkrete Annotationsformate beinhalten). Dieser Umstand ist gerade im Fall des LINGUISTIC ANNOTATION FRAMEWORK kritisch zu sehen, da das zum normativen Teil der Spezifikation gehörende GRAPH ANNOTATION FORMAT nicht durch eine eindeutig festgelegte Dokumentgrammatik definiert ist, was dessen Einsatz als Austauschformat zuwider laufen könnte (vgl. die Ausführungen in Abschnitt 9.4).

Der zweite angeführte Aspekt der innerhalb der Grammatik deklarierten Elemente und Attribute ist insofern interessant, dass in dieser Hinsicht eine eindeutige Entwicklung weg von spezifischen Annotationsformaten hin zu generischen Auszeichnungen zu verzeichnen ist. Die TEI GUIDELINES setzen den Schwerpunkt auf ein Tagset, das zur Auszeichnung geisteswissenschaftlicher Daten vorrangig textueller Art ausgelegt und gleichzeitig unabhängig von spezifischen Theorien ist. Damit können die Elemente und Attribute als Träger von Informationen (und nicht nur zur Strukturierung) aufgefasst werden – ein Konzept, das üblicherweise mit der Inline-Annotation einhergeht (vgl. auch die Ausführungen in Abschnitt 6.4). Allerdings bedingt die Unabhängigkeit von bestimmten Theorien, dass oft mehrere Varianten zur Verfügung stehen, spezifische Phänomene zu annotieren. Die TEI GUIDELINES enthalten in solchen Fällen Empfehlungen, oftmals liegt die finale Entscheidung für die Verwendung einer konkreten Auszeichnung aber beim Anwender. Man kann argumentieren, dass sich bereits darin ein erster Schritt hin zu einer unspezifischeren Annotation manifestiert. Aber auch bei der Einführung generischer Annotationskomponenten reicht der Einfluss der TEI GUIDELINES weit zurück: Angefangen von der P1 (vgl. Listing 6.1 auf Seite 147) über die Entwicklung des GENERIC MAPPING TOOL (GMT, vgl. auch Abschnitt 4.2) und dessen Einsatz in XCES bis hin zu früheren Fassungen des LINGUISTIC ANNOTATION

[4] Als „Definition“ wird in diesem Zusammenhang die Bereitstellung eines vollständigen Schemas verstanden. Das in ISO 12620:2009 enthaltene DATA CATEGORY INTERCHANGE FORMAT (DCIF) wird in dieser Betrachtung nicht berücksichtigt, da es nicht der Annotation linguistischer Primärdaten dient.

FRAMEWORK (vgl. Ide und Romary 2007) ähneln sich die Serialisierungsformate stark. Ein Grund für diese frühe Aufnahme generischer Strukturen in das TEI-Tagset liegt darin begründet, dass die präskriptive Form klassischer Annotationsformate trotz der genannten Varianten oft als (1) nicht adäquat oder (2) nicht präzise genug empfunden wird, um alle auszuzeichnenden Phänomene zu beschreiben. Die Bereitstellung eines relativ übersichtlich gehaltenen Annotationsinventars in Form der standardisierten Beschreibung von Merkmalsstrukturen erlaubt den universellen Einsatz, auch für die Auszeichnung linguistischer Inhalte (vgl. Witt, Rehm *et al.* 2009; Stegmann und Witt 2009) und entkräftet somit den ersten Kritikpunkt. Daher ist es nur schlüssig, dass auch die weiteren in dieser Arbeit diskutierten Normen LINGUISTIC ANNOTATION FRAMEWORK, MORPHO-SYNTACTIC ANNOTATION FRAMEWORK und SYNTACTIC ANNOTATION FRAMEWORK darauf zurückgreifen.

Diese Form der Auszeichnung birgt aber auch die im zweiten Kritikpunkt genannten Risiken: Eine Gegenüberstellung der beiden Listings 16.1 und 16.2 mit Listing 16.3 verdeutlicht das Problem der fehlenden Präzision.[5] Während ersteres zur Benennung der Merkmale und Werte deutsche Begriffe verwendet, setzt Listing 16.2 dafür englische Termini ein (z. B. „vorname“ vs. „firstname“).

Listing 16.1: Merkmalsstrukturbeschreibung (Variante 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="iso-fs-standalone.rnc" type="application/relax-ng-compact-syntax"?>
<fs type="person">
 <f name="vorname">
  <string>max</string>
 </f>
 <f name="nachname">
  <string>meier</string>
 </f>
 <f name="geschlecht">
  <symbol value="männlich"/>
 </f>
</fs>
```

Listing 16.2: Merkmalsstrukturbeschreibung (Variante 2)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="iso-fs-standalone.rnc" type="application/relax-ng-compact-syntax"?>
<fs type="person">
 <f name="firstname">
  <string>max</string>
 </f>
 <f name="surname">
  <string>meier</string>
 </f>
 <f name="gender">
  <symbol value="male"/>
 </f>
</fs>
```

[5] Die Listings 16.1 und 16.2 sind Varianten des Listings 6.2, Listing 16.3 ist als Listing 6.3 unverändert bereits in Abschnitt 6.1 aufgeführt.

Sowohl Listing 16.1 als auch Listing 16.2 sind valide Instanzen nach der Dokumentgrammatik zur Beschreibung standardisierter Merkmalsstrukturbeschreibungen. Durch die unterschiedliche Benennung der Merkmale sind die Informationen aber nicht ohne weitere Verarbeitung vergleichbar.

Listing 16.3: Klassische Inline-Darstellung

```
<?xml version="1.0" encoding="UTF-8"?>
<person geschlecht="männlich">
 <vorname>max</vorname>
 <nachname>meier</nachname>
</person>
```

Dagegen sind die in Informationen in Listing 16.3 (mit Ausnahme des Wertes des Attributs `geschlecht` – sofern mögliche Werte nicht über eine Auswahlliste in der Dokumentgrammatik festgelegt sind) durch eine Dokumentgrammatik limitiert und daher leicht miteinander in Beziehung zu setzen. Die generischen Formate sind im Vergleich formal gesehen eleganter und universeller einsetzbar, dies aber auf Kosten der Präzision (Bański und Przepiórkowski 2010a, S. 98f.):

> There is a family of ISO standards developed by ISO TC 37 SC 4 for modelling and representing different types of linguistic information. The two published standards concern the representation of feature structures (ISO 24610-1) and the encoding of dictionaries (ISO 24613). Other proposed standards are at varying levels of maturity and abstractness. While eventually these standards may reach stability and specificity required by practical applications, this is currently not the case [...] A tendency may be observed of increasing abstractness and generality of proposed standards, esp., SynAF (ISO 24615) and LAF (ISO 24612). This leads to their greater formal elegance, at the cost of their actual usefulness.

Aus diesem Grund verknüpfen die in dieser Arbeit diskutierten Normen auch das Konzept der generischen Annotation mit einer Registrierung der verwendbaren Datenkategorien (DCR, vgl. Kapitel 8). Diese übernimmt damit eine disambiguierende, präzisierende Funktion und fügt darüber hinaus noch einen semantischen Anteil hinzu, da die Datenkategorien nicht nur in Relationen zueinander angeordnet werden können, sondern auch mit weiteren Informationen angereichert werden, was die Präzision der Auszeichnung erhöhen kann.

16.1.3 Formales Modell

Die Sichtweise des formalen Modells von XML-Instanzen als rein Baum-basierte Datenstruktur wurde bereits in Abschnitt 3.3.5 diskutiert und derart modifiziert, dass bei Verwendung einer geeigneten Dokumentgrammatik auch ausdrucksstärkere Modelle repräsentiert werden können. Diese Entwicklung spiegelt sich auch in den formalen Modellen der in dieser Arbeit diskutierten Standards wider: Baum-basierte Ansätze, wie sie in den TEI GUIDELINES vorherrschen, werden von Graphen-basierten mehr

und mehr abgelöst.[6,7] Hier ist vorrangig der CORPUS ENCODING STANDARD, vor allem aber XCES in der Fassung von 2003 zu nennen. Gerade diese Version von XCES nutzt durch die Verwandtschaft zum GENERIC MAPPING TOOL den Graphen als Datenmodell in der Annotation linguistischer Korpora, beide Spezifikationen kennen aber auch weiterhin Baum-basierte Konstruktionen. Auch die aktuellen in ISO/TC 37/SC 4 entwickelten Normen verwenden jeweils einen gerichteten azyklischen Graphen (DAG, vgl. Abschnitt 3.3.3), der um Annotationen erweitert wird (die sich ihrerseits wiederum als Graphen darstellen lassen, Bański und Przepiórkowski 2010b, S. 36). Motiviert wird diese Entwicklung dadurch, dass multiple Annotationen immer stärker in den Fokus linguistischer Forschung gerückt werden (hier seien beispielsweise die Arbeiten des Projekts „Sekimo“ genannt).

16.1.4 Trennung von Konzept und Serialisierung

Die Trennung von Konzept und Serialisierung (zwischen Level und Layer im Sinne des Abschnitts 3.5) ist eng verbunden mit der Notation: Inline-Annotation resultiert fast zwangsläufig mit einer Einheit von Konzept und Serialisierung, hier steht der Element- oder Attributname üblicherweise stellvertretend für das Konzept, das mit der Auszeichnung verbunden wird. Eine Ausnahme von dieser Regel stellen die sogenannten MICROFORMATS dar (Suda 2006; Allsopp 2007; Beer 2008; Lewis 2009; Maurice 2009). Diese aus XHTML bekannte Notationsform nutzt die in XHTML vorhandenen generischen Elemente wie `div` oder `span` in Kombination mit dem Universal-Attribut `class` zur weiteren Spezifizierung des Elementinhalts. Dabei werden die möglichen Werte des `class`-Attributs im Sinne einer Datenkategorieauswahl für bestimmte Anwendungen eingeschränkt, so dass eine Austauschbarkeit und maschinelle Verarbeitung möglich ist.[8] Beispiel für MICROFORMATS wären HCARD für Visitenkarten-Informationen oder XFN, das XHTML FRIENDS NETWORK.[9] Im Vergleich zu spezifischer Inline-Annotation auf der einen und generischer Annotation auf der anderen Seite lassen sich MICROFORMATS als Zwischenstufe der Entwicklung einordnen.[10]

[6] Die Aussage soll nicht so verstanden werden, dass das Tagset der TEI GUIDELINES nicht auch zur Kodierung von Graphen verwendet werden kann – im Gegenteil: Neben den Feature Structures gibt es auch ein Annotationsinventar zur Beschreibung von Graphen in P5 1.9.1, Kapitel 19, „Graphs, Networks, and Trees“. Primär zielt die Auszeichnungssprache aber weiterhin auf Inline-Annotation und damit auf Baum-basierte Strukturen ab.

[7] Die Ablösung des Baums als dominierendes formales Modell kann auch damit belegt werden, dass neuere auf XML-basierende Spezifikationen wie XML SCHEMA, XPath und XQUERY nicht mehr auf der Baum-Repräsentation einer XML-Instanz aufsetzen, sondern vielmehr auf dessen XML INFORMATION SET, das deutlich abstrakter ist. Diese Tatsache wird oftmals verdrängt; ein Umstand, der in Wilde und Glushko (2008, S. 42) als „Tree trauma“ bezeichnet wird.

[8] Im Element `a` zur Auszeichnung von Links und Ankern kann darüber hinaus über das `rel`-Attribut eine unidirektionale Beziehung zum durch das `href`-Attribut definierten Ziels ausgedrückt werden. Durch das `rev`-Attribut kann eine Relation in umgekehrter Richtung realisiert werden.

[9] Nähere Informationen zum XFN finden sich in Maurice 2009, S. 71ff. und unter der URL `http://gmpg.org/xfn/`, zu Microformats allgemein unter der URL `http://microformats.org/`, beide zuletzt abgerufen am 19.04.2012.

[10] Technisch gesehen bleiben MICROFORMATS bis zum Abschluss der Entwicklung von HTML5 eine Zwischenlösung, da sich das Annotationsinventar von HTML erst mit dieser neuen Version hin ändert.

Mit Etablierung der Standoff-Notation (vgl. Abschnitt 16.1.1) wird die Unterscheidung von Konzept und Serialisierung erleichtert. Auch hier lässt sich eine zeitliche Entwicklung beobachten: Im Gegensatz zu den TEI GUIDELINES, die Standoff-Annotation vorrangig für multiple Hierarchien empfehlen, propagieren CES und XCES diese zwar, trennen allerdings nicht grundsätzlich zwischen Konzept und Serialisierung (erst das in der späteren Fassung von XCES enthaltene GMT erlaubt die stärkere Abstraktion). Diese Trennung manifestiert sich endgültig bei der standardisierten Beschreibung der Merkmalsstrukturen in ISO 24610-1:2006 und wird durch die Normen zur Beschreibung des LINGUISTIC ANNOTATION FRAMEWORK, des MORPHO-SYNTACTIC ANNOTATION FRAMEWORK und des SYNTACTIC ANNOTATION FRAMEWORK und deren verbindliche Verwendung einer Datenkategorisierungsregistrierung, sprich: der Externalisierung der konzeptuellen Ebene, forciert.

Zusammengefasst lassen sich die bisherigen Ergebnisse in Tabelle 16.1 zusammenfassen.

Tabelle 16.1: Übersicht der diskutierten Spezifikationen

	TEI	FS	CES	XCES	LAF/GRAF	MAF	SYNAF
Syntax	SGML/XML	XML	SGML	XML			
Notation	Inline/Standoff	Standoff					
Formales Modell	Baum	Graph	Baum/Graph		Graph		
CL	RNG[1]	RNG	DTD	XSD	×	RNG	×
Grammatikklasse		LTG			×	LTG	×
Ebenen	×	✓	×	(✓)	✓	✓	✓

✓/× – Merkmal vorhanden/nicht vorhanden.

[1] Aktuelle Version P5: zusätzlich auch DTD, XSD. P3: SGML-DTD; P4: XML-DTD.

Dass die aktuellen in ISO/TC 37/SC 4 entwickelten Normen ihre Wurzeln in der TEI haben, bestätigen auch Bański und Przepiórkowski (2010b, S. 36):

> The current standards that have been or are being established by ISO TC 37 SC 4 committee (http://www.tc37sc4.org/), known together as the LAF (Linguistic Annotation Framework) family of standards, [...] descend in part from an early application of the TEI, back when the TEI was still an SGML-based standard. That application was the Corpus Encoding Standard [...], later redone in XML and known as XCES [...]. XCES was a conceptual predecessor of the current ISO LAF pivot format for syntactic interoperability of annotation formats, GrAF (Graph Annotation Framework) [...]. GrAF defines an XML serialization of the LAF data model consisting of directed acyclic graphs with annotations (also expressible as graphs), attached to nodes. This basic data model is in fact common to the TEI formats defined for the NCP, the LAF family of standards, and the other standards and best practices such as Tiger-XML [...] – popular for tree-bank encoding, or PAULA [...] – a versatile format for multi-modal and multi-layered corpus encoding.

16.2 Interaktion von Normen untereinander

Anmerkung Die im Folgenden diskutierten Relationen beziehen sich auf die in Teil II genannten Versionen der Spezifikationen. Es ist durchaus möglich, dass einige der hier aufgeführten Kritikpunkte in neueren Fassungen bereits adressiert wurden, diese dem Verfasser aber nicht vorliegen.

Wie aus den Einzelbetrachtungen der Normen ersichtlich, sind einige der Internationalen Standards eng miteinander verknüpft. Die grundlegenden Abhängigkeiten sind naturgemäß bereits durch die verwendete Metasprache XML geschaffen, d. h. Syntax und mögliche Constraint Languages sind vorgegeben. Ebenso basal ist die Referenz auf Unicode als Zeichensatzsystem. Darüber hinaus bestehen aber noch weitere Relationen sowohl zwischen den Normen, die ein formales Modell und ein dazugehöriges Tagset definieren, als auch zwischen den so modellierten Auszeichnungssprachen und der in ISO 12620:2009 standardisierten Datenkategorienregistrierung.

Der Bezug auf andere Internationale Standards ist üblicherweise zu Beginn einer Norm im Abschnitt „Normative references“ aufgeführt, teilweise kann es aber vorkommen, dass auch an anderen Stellen der Spezifikation Verweise aufgeführt werden – ein Beispiel dafür wäre die in Abschnitt 11.2 vorgestellte Spezifikation des SYNTACTIC ANNOTATION FRAMEWORK. Zwei grundlegende Beziehungen lassen sich festmachen: Zum einen stellt die in ISO 24610-1:2006 standardisierte Beschreibung von Merkmalsstrukturen die Basis für die konkreten Serialisierungsformate der anderen Normen. Zum anderen stellt die in ISO 12620:2009 definierte Datenkategorienregistrierung eine Speicherungsmöglichkeit für die semantischen Konzepte der anderen Standards zur Verfügung, so dass in diesen nur eine Kategorieselektion (DCS) erstellt werden muss.

Das LINGUISTIC ANNOTATION FRAMEWORK erweitert die in ISO 24610-1:2006 normierten Merkmalsstrukturbeschreibungen: Es expliziert das formale Modell des Graphen durch entsprechende Elemente im GRAF und verwendet Merkmalsstrukturen zur Speicherung der konkreten Annotationen. Ergänzt werden diese Komponenten durch die in der Spezifikation aufgeführten Metadaten in Form der jeweiligen Header, die sich wiederum am CORPUS ENCODING STANDARD bzw. den TEI GUIDELINES orientieren. Allerdings übernimmt die aktuelle Fassung des Standards (ISO/FDIS 24612) nicht vollständig das Annotationsformat zur Beschreibung von Merkmalsstrukturen, sondern variiert die Serialisierung: Statt für die Darstellung der Feature Structures direkt auf ISO 24610-1:2006 zu verweisen, ist eine eigenständige, minimal kürzere Notation Teil der Spezifikation (vgl. Kapitel 9.4). Dabei wäre aus Gründen der Interoperabilität die Einbindung in GRAF (die in technischer Hinsicht über XML NAMESPACES erfolgen kann) sinnvoller.[11] Alternativ dazu wäre die Modellierung des GRAF-eigenen Elements `fs` (also der die Merkmalsstruktur repräsentierende Teilbaum) analog zu der in ISO 24610-1:2006 denkbar. Es bleibt abzuwarten, ob hier im Rahmen der weiteren Normungsarbeit noch auf einen stärkeren wechselseitigen Bezug der Standards hingearbeitet wird.

Das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK (vgl. Kapitel 10) sowie das SYNTACTIC ANNOTATION FRAMEWORK (vgl. Kapitel 11) stellen im weiteren Sinne Konkretisie-

[11] Wie eine solche Einbettung erfolgen kann, ist in Teil III demonstriert. Der weitere Aufbau, inkl. der Darstellung der Knoten und Kanten sowie der Metadaten in GRAF bliebe davon unberührt – wie es in XSTANDOFF auch der Fall ist.

rungen des allgemeineren LAF dar. Auch hier sind die Beziehungen aber nicht eindeutig festgelegt. Zwar werden in ISO/DIS 24611, Abschnitt 4, „Key standards used by LMF“[12] neben OLAC-Metadaten (vgl. Abschnitt 12.2) und UML (ISO/IEC 19501:2005; ISO/IEC DIS 19505-1; ISO/IEC DIS 19505-2; OMG 2011a; OMG 2011b) sowohl ISO 24610-1:2006 als auch ISO 12620:2009 aufgeführt. Allerdings bezieht sich der in ISO/DIS 24611, S. 35 zu findende Verweis auf das Linguistic Annotation Framework nur auf die Festlegung eines Adressierungsschemas für die Standoff-Notation. Dafür wird für die Annotation des morphosyntaktischen Inhalts auf ISO 24610-1:2006 zurückgegriffen (vgl. auch Listing 10.1 auf Seite 217). Hier sollte noch vor Verabschiedung der endgültigen Fassung ein Serialisierungsformat zur Beschreibung von Merkmalsstrukturen verbindlich empfohlen werden.

Ähnlich sind die Verhältnisse beim Syntactic Annotation Framework. Die bereits als Internationaler Standard ISO 24615:2010 verabschiedete Spezifikation bezieht sich im Vorwort explizit auf die Arbeiten im Rahmen des Linguistic Annotation Framework:

> ISO 24615 is designed to coordinate closely with ISO CDI 24612 Language resource management – Linguistic annotation framework (LAF), and ISO 24613: 2008 Language resource management – Lexical markup framework (LMF), and ISO CD 24611 Language resource management – Morphosyntactic Annotation Framework (MAF). (ISO/FDIS 24615, S. iv)

Auch enthält diese letzte frei verfügbare Fassung einen informativen Anhang B „Relation to the Linguistic Annotation Framework“, der aber wenig konkret bezüglich der Verbindungen zwischen beiden Normen bleibt. Es ist daher unklar, ob die Strukturierung der Annotationen gemäß ISO 24610-1:2006 oder nach der davon teilweise abweichenden GrAF-Notation zu erfolgen hat.

Diese nicht-eindeutigen Relationen sind der Tatsache geschuldet, dass die zeitlichen Abläufe der Normierung der einzelnen Standards voneinander abweichen, so dass gegenseitige Abhängigkeiten nicht immer korrekt aufgelöst werden können. Problematisch ist hier vorrangig das Graph Annotation Format, dessen Ausprägung sich in den verschiedenen Entwicklungsstadien mehrfach in größerem Umfang geändert hat, und in der letzten Fassung der Spezifikation (ISO/FDIS 24612) gar nicht mehr als konkretes Annotationsformat, d. h. in Form einer Dokumentgrammatik Teil des Standards ist.[13] Diese Problematik wird weiter dadurch verschärft, dass seit einiger Zeit unterschiedliche und teilweise untereinander nicht kompatible Fassungen des Graph Annotation Format verwendet werden – und damit ungewollt der Zweck eines Austauschformats konterkariert wird (vgl. Abschnitt 9.3).

Ein denkbarer Lösungsansatz wäre, in der noch nicht verabschiedeten Norm zur Spezifikation des Morpho-syntactic Annotation Framework – ähnlich wie im

[12] Bei der Abschnittsüberschrift scheint es sich um einen klassischen Copy-Paste-Fehler zu handeln.

[13] Zu beachten ist, dass GrAF auch in den vorherigen Versionen immer Teil des informativen Anhangs war, so dass das XML-Schema nur einen unverbindlichen Charakter hatte. Allerdings ist auch in der aktuellen Fassung der Norm (ISO/FDIS 24612, S. 5) weiterhin die folgende Aussage zu finden (Hervorhebung durch den Verfasser): „This framework includes an abstract data model *and an XML serialization of that model* for representing annotations of primary data.“

Falle von ISO 24615:2010 – auf ein konkretes Serialisierungsformat zu verzichten und direkt auf GrAF zu verweisen. Dazu wäre allerdings eine (im besten Falle normative) GrAF-Serialisierung in Form einer Dokumentgrammatik erforderlich, die in der aktuellen Fassung ISO/FDIS 24612 fehlt. Auch in diesem Sinne wäre der oben bereits angesprochene Import des RELAX-NG-Schemas für die standardisierte Merkmalsstrukturbeschreibung aus ISO 24610-1:2006 bzw. den TEI Guidelines hilfreich, da so Feature-Structure-Repräsentationen recht variabel austauschbar wären: entweder alleinstehend im Sinne von ISO 24610-1:2006, als Teil einer TEI-Instanz oder als Teil einer GrAF-Instanz zur Beschreibung allgemeiner linguistischer Strukturen (im Sinne des Linguistic Annotation Framework) oder syntaktischer bzw. morphosyntaktischer Merkmale (im Sinne des Syntactic Annotation Framework bzw. Morpho-syntactic Annotation Framework).

Abbildung 16.1 versucht die genannten Relationen der einzelnen Normen untereinander bzw. zu den weiteren diskutierten Standards grafisch zusammenzufassen.[14]

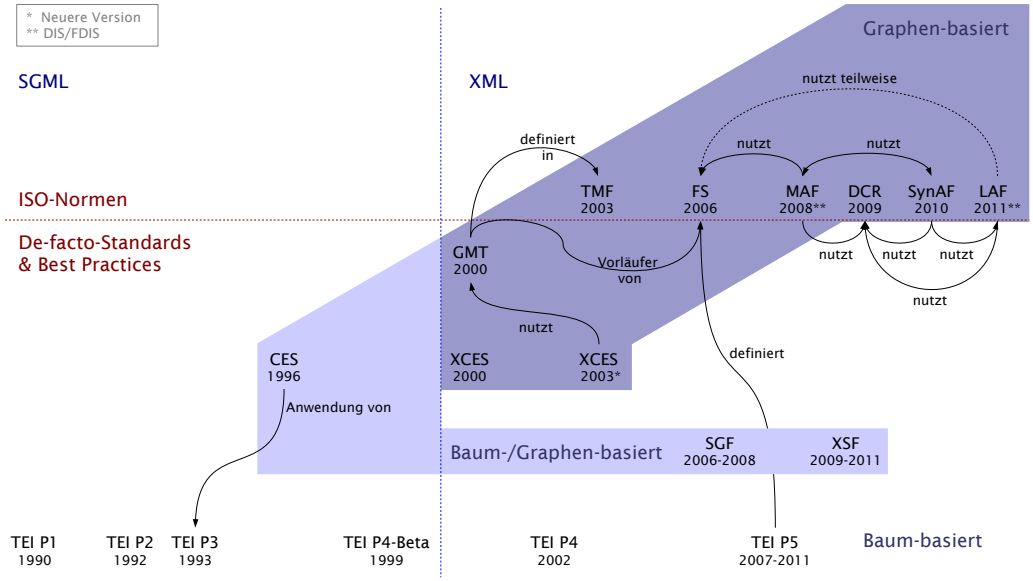


Abbildung 16.1: Relationen der in Teil II diskutierten Spezifikationen

Der zeitliche Verlauf der Entwicklung orientiert sich an der (nicht-dargestellten) X-Achse. Zunächst können die Spezifikationen anhand ihrer Metasprache (SGML, linke Hälfte vs. XML, rechte Hälfte) und ihres Status' differenziert werden (Normen und deren Vorläufer in der oberen Hälfte, sonstige Spezifikationen in der unteren Hälfte). Darüber hinaus wird das zu Grunde liegende formale Modell farblich verdeutlicht: Der weiße Bereich mit den TEI Guidelines umschließt vorrangig Baum-basierte Standards, der dunkel-violette Bereich Graphen-basierte. Da sowohl der Corpus Encoding Standard als auch SGF und XStandoff Mischformen darstellen, wurde der sie umfassende Bereich entsprechend blass-violett gefärbt.

[14] Die Abbildung enthält im Sinne einer Vereinfachung nur die wesentlichen Beziehungen, die alleinige Nennung einer Norm innerhalb einer anderen wird nicht dargestellt.

Zur besseren Darstellung der Entwicklung der standardisierten Beschreibung von Merkmalsstrukturen wurden das GENERIC MAPPING TOOL und die dieses beinhaltende Norm ISO 16642:2003 mit in das Diagramm aufgenommen. Da eine Dokumentgrammatik für das GMT zwar in der frühen Fassung ISO/CD 12620-1 der späteren Norm ISO 12620:2009 enthalten ist, aktuelle Schemata für das DATA CATEGORY INTERCHANGE FORMAT (DCIF) allerdings erheblich davon abweichen, wurde auf eine Beziehung zwischen diesen beiden Spezifikationen verzichtet. Ebenfalls fehlt aus Gründen der Übersichtlichkeit die grafische Darstellung der Relation zwischen früheren Fassungen der TEI GUIDELINES und der ISO 24610-1:2006. Wie in Abschnitt 6.1 dargelegt, sind Merkmalsstrukturbeschreibungen bereits lange Bestandteil der TEI GUIDELINES.[15] Zu beachten ist, dass die Relation zwischen den Spezifikationen MAF und SYNAF bidirektionaler Art ist, da beide Standards sich gegenseitig aufeinander beziehen.

16.3 XStandoff im Vergleich

Das in Teil III vorgestellte und vom Verfasser dieser Arbeit entwickelte XSTANDOFF ist ebenfalls von verschiedenen anderen Spezifikationen beeinflusst worden. Direktes Vorbild ist die bereits in der ersten Phase des Projekts „Sekimo“ erarbeitete Prolog-Faktenbasis (vgl. Abschnitt 3.3.2.4), aber auch andere im gleichen Zeitraum entwickelte Formate, wie das in Abschnitt 4.2 bereits angerissene POTSDAMER AUSTAUSCHFORMAT FÜR LINGUISTISCHE ANNOTATIONEN (PAULA) haben Auswirkungen auf die ersten Versionen von SGF und XSTANDOFF gehabt. Bezüglich der in Abschnitt 16.1 diskutierten vier Komponenten einer Auszeichnungssprache lässt sich XSTANDOFF wie folgt einordnen: Als Hybrid-Ansatz verwendet das Format die Standoff-Notation unter Einbehaltung der hierarchischen Beziehungen zwischen Elementen, so dass Teilbäume vollständig erhalten bleiben (abzüglich der Blattinformationen, sprich: der Primärdaten) und folgt damit der in Abschnitt 16.1.1 skizzierten Entwicklung. Im Gegensatz zu den Ergebnissen des Abschnitts 16.1.2 ist das XSTANDOFF definierende Schema unter formalen Gesichtspunkten als Unambigious Restrained Competition Grammar (URCG) einzuordnen, da es sowohl Wildcards als auch Co-Constraints einsetzt. Darüber hinaus ist die Verwendung von XML SCHEMA auch aus technischen Gründen motiviert; hier sei auf die ausgereifte Unterstützung von XML NAMESPACES, der gegenüber RELAX NG besseren Integritätsmechanismen und der genaueren Steuerung des Parsingverhaltens für Teilbäume (vgl. Abschnitt 13.4.1) hingewiesen. In Hinsicht auf das formale Modell kann XSTANDOFF aufgrund der Unterstützung für diskontinuierliche Annotationseinheiten die Abbildung von Generalized GODDAGs bescheinigt werden, dies kann als logische Fortführung der in Abschnitt 16.1.3 skizzierten Entwicklung angesehen werden. Konzept und Serialisierung werden explizit durch entsprechende Elemente in der Instanz separiert. Bei Bedarf kann zusätzlich eine Referenz auf eine DCS bzw. konkrete Kategorien innerhalb einer Implementierung wie ISOCAT durch das Universal-Attribut `dcrCategory` einzelnen Knoten bzw. Teilbäumen hinzugefügt werden, so dass auch hier semantische Informationen vollständig an eine externe Datenkategorienregistrierung ausgegliedert

[15] Der Vollständigkeit halber müssten Verknüpfungen zwischen den ersten Fassungen der TEI GUIDELINES, GMT und ISO 24610-1:2006 eingezeichnet werden.

werden können.

Im Unterschied zu den in Teil II diskutierten Standards erlaubt XStandoff darüber hinaus weitreichende Freiheiten bzgl. der Strukturierung von Informationen sowie die Kombination bestimmter Serialisierungsformate mit spezifischen Metadaten-Standards (beispielsweise TEI-Merkmalsstrukturen und OLAC-Metadaten). Dabei wird der Ansatz verfolgt, möglichst viele Informationen in einer Instanz zu bündeln, was prinzipiell alle ein spezifisches Primärdatum betreffenden Annotationsebenen, aber auf Wunsch auch dieses selbst einschließt. Darin liegt einer der wesentlichen Unterschiede dieses Metaformats, während beispielsweise in ISO/FDIS 24612, S. 9 (ähnlich wie in XCES oder PAULA) die Speicherung in separaten Instanzen empfohlen wird: „LAF recommends representing each of the linguistic layers defined in ISO/TC 37/SC 4, Language resource management, in a separate annotation document for the purposes of exchange." Der Ansatz von XStandoff geht dagegen so weit, dass auch ganze Korpora in einer Instanz zusammengefasst werden können (dann sinnvollerweise nicht mehr als Datei im Dateisystem, sondern vorzugsweise in einer nativen XML-Datenbank, vgl. Abschnitte 15.3 und 13.5). Alternativ können Korpuseinträge (`corpusData`-Teilbäume) in externe Dateien ausgelagert und über das `corpusDataRef`-Element referenziert werden. Analog dazu können Metadaten in eigenen Dateien gespeichert werden, nicht aber einzelne Annotationsebenen, da in diesem Fall die Integrität nicht gewährleistet werden kann. Weitere Unterschiede beziehen sich auf das Segmentierungsschema: ISO/FDIS 24612 verwendet Anker, die in Abhängigkeit der Primärdaten und der jeweiligen Implementierung definiert sind (die Spezifikation bliebt hier leider ungenau). XCES bezieht sich für die Segmentierung auf die XML Pointer Language (vgl. Abschnitt 7.4), die nicht mehr weiterentwickelt wird, während die TEI Guidelines die Nachfolge-Spezifikation XPointer verwenden (vgl. Abschnitt 5.6). ISO/DIS 24611 enthält keine konkreten Angaben zur Segmentierung bei Verwendung der Standoff-Notation (vgl. Abschnitt 10.1.1), gleiches gilt für ISO/FDIS 24615. XStandoff unterstützt eine Reihe von Segmentierungsschemata (vgl. Abschnitt 13.2.2) und ist in dieser Hinsicht autonom gegenüber anderen Spezifikationen.

16.3.1 Weitere Arbeiten

Unabhängig von den Entwicklungen in ISO/TC 37/SC 4 gibt es weitere Annotationsformate, die XStandoff ähneln: Eckart und Teich 2007 demonstrieren mit Generalised Annotation Markup (GAM) ein Format, das ebenfalls multiple Annotationseinheiten in einer einzelnen Instanz speichert und in Kombination mit der Datenbank AnnoLab eingesetzt wird (Eckart 2008).[16] Zur Segmentierung werden einfache numerische Angaben verwendet, analog zum Vorgehen in XStandoff, insofern ähneln sich die beiden Formate in einigen Punkten.

Eine aktuelle ähnliche Spezifikation stellt das im Projekt „D-SPIN" (aktuell: „CLARIN-D") entwickelte Text Corpus Format (TCF, Heid *et al.* 2010) dar. Auch dieses fasst

[16] AnnoLab setzt auf die native XML-Datenbank eXist auf. Eine ältere Version kann unter `http://www.annolab.org/` heruntergeladen werden, zuletzt abgerufen am 19.04.2012. Dem Anschein nach werden weder AnnoLab noch GAM aktiv weiterentwickelt.

multiple Annotationsebenen in einer Instanz zusammen. Obwohl grundlegend Standoff-basiert, können Token auch inline annotiert werden (vgl. Abbildung 2, S. 495 Heid *et al.* 2010), was die Auszeichnung diskontinuierlicher oder überlappender Einheiten verhindert.[17] TCF ist als Repräsentationsformat für mehrfach annotierte Korpora vorgesehen und soll zwischen der Ausgabe verschiedener linguistischer Werkzeuge und dem normierten GrAF dienen. Dieser Zwischenschritt, der eine zusätzliche Konvertierung erfordert, bleibt allerdings etwas unmotiviert.

16.3.2 Alternative Kriterien

Berücksichtigt man, dass sowohl die Verwendung des Graphen als Datenmodell als auch der Einsatz der Standoff-Notation ihren Ursprung im Wunsch nach der Abbildung multipler Hierarchien haben, ist es nur folgerichtig, die untersuchten Annotationsformate in Hinblick darauf zu evaluieren. DeRose (2004, S. 3) hat dazu eine Reihe von Kriterien aufgestellt, die im weiteren Sinne auch jetzt noch als gültig anzusehen sind:[18]

1. Lesbarkeit für den Menschen.
2. Vorhandene Implementierungen.
3. Aufwand zur Erzeugung multipler Sichtweisen.
4. Aufwand zur Extraktion einzelner Hierarchien.
5. Wartbarkeit.
6. Einfachheit der Notation.
7. Einfachheit der Validierung.
8. Möglichkeit der Cross-Layer-Validierung.
9. Integrität der Primärdaten in bzgl. deren Abfolge.

Der Aspekt der Lesbarkeit für den Menschen ist bei allen Standoff-basierten Formaten problematisch, da im Gegensatz zu Inline-Annotationen der entsprechende Abschnitt des Primärdatums nicht direkt sichtbar ist, sondern nur über eine Referenz ermittelt werden kann. Zu diesem Zweck enthält XStandoff das optionale `content`-Attribut unterhalb des `segment`-Elements, das zur Speicherung des durch das Segment umspannten Bereichs der Primärdaten genutzt werden kann.[19] Darüber hinaus erlaubt die Erhaltung der XML-inhärenten Baumstruktur (unterhalb eines `layer`-Elements) in

[17] Aus Heid *et al.* (2010) geht nicht eindeutig hervor, ob die Annotation der Token alternativ auch in Standoff-Notation erfolgen kann, es sollte aber davon ausgegangen werden.

[18] Die Reihung der Kriterien wurde angepasst, der in DeRose (2004) angeführte Punkt „Kompatibilität zu XML“ wird aus nachvollziehbaren Gründen nicht berücksichtigt.

[19] Natürlich muss auch in diesem Fall zunächst ausgehend von einer bestimmten Annotation das entsprechend referenzierte Segment herangezogen werden. Die Auflösung dieser Referenz kann aber sehr einfach über die `idref()`-Funktion in XPath 2.0 erfolgen (vgl. Malhotra *et al.* 2010, Abschnitt 15.5.3, „fn:idref“).

XSTANDOFF zumindest XML-affinen menschlichen Nutzern die schnelle Erkennung von Knoten-Relationen. Zwar können solche hierarchischen Beziehungen auch durch komplexe Merkmalsstrukturen dargestellt werden, sind aber durch die Umwandlung in eine generische Annotation nicht mehr so offensichtlich (vgl. die Listings 16.1 und 16.3). Darüber hinaus kann diese Problematik durch geeignete Software zur Bearbeitung von Standoff-annotierten Daten umgangen werden.[20] Beispiel dafür sind die bereits in Abschnitt 16.1.1 angeführten Implementierungen SERENGETI und die GLOZZ ANNOTATION PLATFORM. Für fast alle der in Teil II diskutierten ISO-Normen fehlen spezifische Implementierungen. Positive Ausnahmen sind hier das <tiger2/>-Format als mögliches Serialisierungsformat von Informationen, die mittels des MORPHO-SYNTACTIC ANNOTATION FRAMEWORK und SYNTACTIC ANNOTATION FRAMEWORK annotiert wurden, sowie ISOCAT als Web-basiertes Werkzeug zur Erstellung und Pflege von Datenkategorien nach ISO 12620:2009.[21] Die TEI GUIDELINES dagegen wird von einer großen Anzahl an Software unterstützt. Neben Editoren zählen hierzu auch ROMA und VESTA zur Assistenten-gestützten Modifikation der Dokumentgrammatik und die TEI-XSLT-Stylesheets. Keiner der sonstigen in dieser Arbeit genannten Standards kann auf ein derart reichhaltiges und gut gepflegtes Angebot zurückgreifen, was auch direkte Auswirkungen auf die Beurteilung der beiden Punkte „Aufwand zur Erzeugung multipler Sichtweisen“ und „Aufwand zur Extraktion einzelner Hierarchien“ hat. Da die in ISO/TC 37/SC 4 entwickelten Normen Annotationsebenen in separaten Instanzen speichern, sind beide Aspekte identisch zu dem Aufwand, der für die Erstellung bzw. Extraktion einzelner Annotationsebenen erforderlich ist. Sowohl das MORPHO-SYNTACTIC ANNOTATION FRAMEWORK als auch das SYNTACTIC ANNOTATION FRAMEWORK sind konzeptionsbedingt auf die Annotation einer einzelnen Betrachtungsebene ausgelegt, weshalb an diesem Punkt nicht weiter darauf eingegangen werden muss. Ähnliches gilt für die TEI GUIDELINES: In der Inline-Annotation dürften multiple Hierarchien aus den bekannten Gründen eher selten ausgezeichnet werden. Das alternativ zur Verfügung stehende Standoff-Markup bzw. die Merkmalsstrukturbeschreibungen verfahren wiederum analog zu den generischen Formaten wie GRAF. Mit den im XSTANDOFF-Toolkit enthaltenen XSLT-Stylesheets können XSTANDOFF-Instanzen mit multiplen Hierarchien relativ komfortabel erstellt (unter der Prämisse, dass jeweils nur zwei Auszeichnungsebenen kombiniert werden können) und einzelne Annotationsebenen extrahiert werden (vgl. Kapitel 14).

Der Aspekt der Wartbarkeit wird mit der Einfachheit der Notation zusammen diskutiert. Hierunter können zum einen die Stabilität der Spezifikation als solche, zum anderen aber auch Aspekte der Bearbeitung einzelner Instanzen, Dokumentation oder auch der Lizenz subsumiert werden. Internationale Normen schneiden naturgemäß in Bezug auf den ersten Punkt sehr gut ab. Im Idealfall sind kritische Aspekte und die Verträglichkeit gegenüber anderen Standards im Prozess der Normierung bereits berücksichtigt worden, so dass auch nachfolgende Fassungen (eine Revision findet üblicherweise fünf Jahre nach Verabschiedung statt) eher weniger umfangreiche Modifikationen am Kern einer Spezifikation vornehmen. Dass im Rahmen dieses Prozesses allerdings

[20] Während bei DeRose (2004) mit „Implementierung“ noch eher die grundlegende Unterstützung durch Software gemeint ist, sind aktuelle Werkzeuge in der Lage, Standoff-Instanzen zu editieren.

[21] Zumindest für das Vorgänger-Format TIGER-XML sind entsprechende Werkzeuge vorhanden. Es kann davon ausgegangen werden, dass diese für <tiger2/> angepasst werden.

durchaus auch tiefgreifende Anpassungen vorgenommen werden, wird am Beispiel des LINGUISTIC ANNOTATION FRAMEWORK deutlich. Darüber hinaus können auch bei verabschiedeten ISO-Normen teilweise gegenseitige Inkompatibilitäten auftreten (vgl. Abschnitt 16.2). Eher heterogen sieht es in dieser Hinsicht bei den De-facto-Standards aus: Während bei XCES vorrangig die unklare Versionierung und die ungewisse Zukunft als problematisch einzustufen sind, können die TEI GUIDELINES als äußerst positives Beispiel für ein nachhaltiges Annotationsformat dienen. Abgesehen von der überaus umfangreichen und frei verfügbaren Dokumentation überzeugt auch die Unterstützung für Anwender der Vorgängerversion (P4).[22] XSTANDOFF als Spezifikation ohne den Hintergrund einer größeren Entwicklergemeinschaft ist im Vergleich dazu ungleich schlechter aufgestellt. Andererseits ist es als Meta-Auszeichnungssprache weniger starken Änderungen unterworfen und hat im Laufe seiner mehrjährigen Entwicklung einen hohen Reifegrad erlangt. Darüber hinaus erlaubt die liberale Lizenz, unter der sowohl die Dokumentgrammatiken als auch das XSFTk verfügbar sind, die Modifikation durch andere Entwickler (vgl. Abschnitt 15.4).

Die Wartbarkeit der XSTANDOFF-Instanzen ist wie bei allen Standoff-basierten Formaten in hohem Maße von der Funktionalität des XSTANDOFF-Toolkits und weiteren Implementierungen wie dem bereits angesprochenen Annotationswerkzeug SERENGETI abhängig.

Die Validierbarkeit entsprechend annotierter Instanzen setzt naturgemäß das Vorhandensein einer kanonischen Dokumentgrammatik voraus. Für die De-facto-Standards TEI GUIDELINES und mit Abstrichen XCES (vgl. Abschnitt 7.8) sind solche vorhanden. Die noch in der Entwicklung befindliche internationale Norm ISO/DIS 24611 führt zwar im informativen Anhang ein RELAX-NG-Schema auf, es ist aber zum aktuellen Zeitpunkt unklar, ob dieses auch Bestandteil der finalen Fassung sein wird. Sowohl SYNAF (ISO/FDIS 24615) als auch die aktuelle Version der Norm zur Beschreibung des LINGUISTIC ANNOTATION FRAMEWORK (ISO/FDIS 24612) lassen eine Dokumentgrammatik vermissen. XSTANDOFF bietet dagegen eine stabile und in mehreren Versionen vorliegende Dokumentgrammatik, die nicht die Validierung einzelner Annotationsebenen, sondern auch die Cross-Layer-Validierung multipler Annotationen ermöglicht (vgl. Abschnitt 13.4.1). Eine solche ist bei den Formaten, die Annotationsebenen auf verschiedene Instanzen verteilen, nicht ohne zusätzlichen Aufwand möglich, da validierende Parser üblicherweise Instanz-basiert arbeiten.

Der Aspekt der Integrität der Primärdaten in Bezug auf die Abfolge der einzelnen Segmente ist bei den Standoff-Formaten generell unproblematisch, da üblicherweise nur lesend auf die Primärdaten zugegriffen wird. Eine Ausnahme bilden der CORPUS ENCODING STANDARD, der eine basale Auszeichnung der Primärdaten vorsieht, sowie die TEI GUIDELINES. Für letztere gilt, dass multiple Hierarchien aufgrund drohender Überlappungen eher selten inline annotiert werden (zumal dies nicht primärer Einsatzzweck der Spezifikation ist). Ansonsten stehen auch hier die Standoff-Variante sowie der Einsatz von Merkmalsstrukturbeschreibungen zur Verfügung.

[22] Die P4 wird nicht nur weiterhin bis zum November 2012 unterstützt, es stehen auch umfangreiche Hilfsmittel zur Migration auf die aktuelle Fassung zur Verfügung.

Bezug nehmend auf die Tabelle 16.1 auf Seite 292, kann XSTANDOFF zusammenfassend den anderen in dieser Arbeit diskutierten Standards wie folgt gegenübergestellt werden:

Tabelle 16.2: Übersicht der diskutierten Spezifikationen (inkl. XStandoff)

	TEI	FS	CES	XCES	LAF/GRAF	MAF	SYNAF	XSF
Syntax	SGML/XML	XML	SGML	XML				
Notation	Inline/Standoff	Standoff						
Formales Modell	Baum	Graph	Baum/Graph		Graph			Baum/Graph
CL	RNG	RNG	DTD	XSD	×	RNG	×	XSD
Grammatikklasse		LTG			×	LTG	×	URCG[1]
Ebenen	×	✓	×	(✓)	✓	✓	✓	✓

✓/× – Merkmal vorhanden/nicht vorhanden.

[1] Die mittels XML SCHEMA 1.0 modellierte Version kann sowohl als RCG als auch als URCG angesehen werden, je nachdem, ob die eingebetteten SCHEMATRON-Assertions als Teil der Dokumentgrammatik gewertet werden oder nicht. Die mittels XML SCHEMA 1.1 modellierte Fassung kann als URCG angesehen werden.

17 Fazit

Ziel dieser Arbeit war die Diskussion und Evaluation aktueller Spezifikationen und Standards (sowohl im Sinne von De-facto- als auch De-jure-Standards) zur Strukturierung linguistischer Korpora. Zu diesem Zweck wurden durch eine konzise Darstellung der formalen Grundlagen Kriterien zur Beurteilung eben dieser Standards auf Basis der vier distinktiven Komponenten von Auszeichnungssprachen – Syntax und Notation, formales Modell, Constraint Language und Trennung von Konzept und Serialisierung – entwickelt: Dabei wurde das von Sperberg-McQueen und Huitfeldt (1999a); Sperberg-McQueen (2002) eingeführte dreigliedrige Modell von Auszeichnungssprachen um die von Bayerl *et al.* (2003); Witt (2004) diskutierte Trennung von konzeptueller Ebene und konkreter Serialisierung erweitert.

Bei der Diskussion der Syntax kann zum einen die grundlegende syntaktische Form, die abhängig von der verwendeten Metasprache (SGML vs. XML) ist, unterschieden werden von der Notation (Inline vs. Standoff). Auch das formale Modell ist in weiten Teilen abhängig von der Metasprache, die zur Modellierung einer Auszeichnungssprache herangezogen wird. Hier konnte gezeigt werden, dass bei der Einstufung des formalen Modells von XML-basierten Auszeichnungssprachen zu unterscheiden ist zwischen wohlgeformten Instanzen und denen, die einer Dokumentgrammatik gehorchen, da diese in Abhängigkeit des verwendeten Grammatikformalismus' über Integritätsmerkmale die Etablierung zusätzlicher, von der grundlegenden Baumstruktur unabhängiger Verbindungen zwischen Knoten zulassen. Ausgehend davon wurden die verbreiten Schemasprachen DTD, XML Schema und RELAX NG evaluiert. Dabei wurde die von Murata, D. Lee, Mani und Kawaguchi (2005); Murata, D. Lee und Mani (2001) eingeführte Taxonomie um die neu entwickelte Klasse der Unambigious Restrained Competition Grammar (URCG) erweitert. Diese Klasse kann für eine weite Spanne von Grammatiken oder Schemata genutzt werden, die auf ambige Strukturen verzichten, was im Schluss die Verarbeitung erleichtern kann. Die im abschließenden Teil des Kapitels 3 diskutierte Unterscheidung von konzeptueller Ebene (Level) und der davon prinzipiell unabhängigen Serialisierung (Level) stellt ein weiteres wichtiges Kriterium zur Beurteilung von

Auszeichnungssprachen dar.

Ebenfalls im Grundlagenteil wurde in Kapitel 2 der Entwicklungsprozess der im daran anschließenden Teil II erörterten Normen und Spezifikationen skizziert, um die Hintergründe einiger Design-Entscheidungen besser nachvollziehen zu können. Die in Teil II diskutierten Spezifikationen lassen sich unterteilen in die De-facto-Standards TEI Guidelines und Corpus Encoding Standard und die internationalen Normen Linguistic Annotation Framework, Morpho-syntactic Annotation Framework und Syntactic Annotation Framework. Darüber hinaus wurden die sowohl in den TEI Guidelines als auch in Form der ISO-Norm ISO 24610-1:2006 spezifizierten Merkmalsstrukturbeschreibungen sowie der Internationale Standard ISO 12620:2009 zur Beschreibung einer Datenkategorisierungsregistrierung diskutiert.

Als alternativer Ansatz wurde in Teil III die vom Verfasser entwickelte Meta-Auszeichnungssprache XStandoff vorgestellt. Obwohl XStandoff Anleihen bei den diskutierten (sowie weiteren) Formaten macht, kann es sowohl in der Frage des formalen Modells als auch im Aufbau als eigenständig betrachtet werden. Dazu kommt, dass XStandoff im Gegensatz zu den in Teil II erörterten Standards eine Dokumentgrammatik besitzt, deren formale Ausdrucksstärke als URCG einzustufen ist, und die damit die der anderen deutlich übersteigt. Auch im Hinblick auf die Unterscheidung von konzeptueller Ebene und Serialisierung nimmt XStandoff eine deutliche Trennung in Form entsprechender Elemente vor und erlaubt darüber hinaus die Kombination spezifischer Auszeichnungselemente mit Referenzen auf eine standardisierte Datenkategorisierungsregistrierung wie ISOcat.

Sowohl die in Teil II diskutierten Spezifikationen als auch XStandoff wurden anschließend in Kapitel 16 bezüglich der vier distinktiven Komponenten von Auszeichnungssprachen aber auch hinsichtlich ihrer Relationen untersucht. Zusammenfassend ist festzuhalten, dass zwei Trends im Vergleich der in dieser Arbeit diskutierten Auszeichnungssprachen zur Annotation linguistischer Korpora deutlich geworden sind: (1) der breite Einsatz von Standoff-Notation und (2) die Nutzung generischer Annotationskonzepte. Beide Entwicklungen lassen sich vom Ansinnen herleiten, multiple Annotationen strukturiert speichern zu können ohne auf die Metasprache XML verzichten zu müssen. Einhergehend mit den Veränderungen bezüglich der Notation lässt sich auch ein Wandel hin zum Graphen als formales Modell verzeichnen.

Im Sinne einer Empfehlung für ein Annotationsformat für linguistische Korpora nimmt die TEI P5 eine entscheidende Rolle ein. Abgesehen davon, dass diese Spezifikation seit ihren Anfängen eine enorme Entwicklung durchlaufen hat und dabei immer weiter angepasst und erweitert wurde, erlaubt die aktuelle Fassung die schon in früheren Versionen kritisierte Komplexität durch Einführung von ODD und Roma erheblich besser zu verwalten. Diese beiden neuen Technologien erlauben eine äußerst granulare Anpassung an bestehende Bedürfnisse – auch wenn noch nicht alle denkbaren Modifikationen möglich sind.[1]

[1] Der Verfasser dieser Arbeit hat im März 2010 die Anfrage auf der TEI-Mailingliste gestellt, ob es möglich ist, das extern deklarierte Attribut `segment` aus dem Namensraum von XStandoff in eine ODD-Instanz einzufügen, um automatisiert eine entsprechende TEI-Dokumentgrammatik zu erzeugen (zum Zwecke der Nutzung von TEI-Instanzen innerhalb von XStandoff). Sebastian Rahtz, der Roma und die XSLT-Stylesheets der Text Encoding Initiative verwaltet, antwortete darauf wörtlich: „Phew.

Zu diesem Schluss kommen auch Przepiórkowski und Bański (2009, S. 250), die die TEI Guidelines mit XCES, frühen Fassungen des Morpho-syntactic Annotation Framework und des SynAF, TIGER-XML und PAULA in Bezug auf mehrfach annotierte Sprachkorpora vergleichen: „We conjecture that – given the stability, specificity and extensibility of TEI P5 and the relative instability and generality of some of the other proposed standards – this approach is currently the optimal way of following corpus encoding standards."

Daher kann die TEI P5 – trotz der durchaus vielversprechenden Arbeiten aus TC 37 – weiterhin als für linguistische Annotationen universell und nachhaltig einsetzbares Auszeichnungsformat angesehen werden. Es bleibt dennoch zu hoffen, dass die in Abschnitt 16.2 bemängelten Bezüge zwischen einzelnen Normen und Normenvorschlägen auf absehbare Zeit adressiert werden. Bis dahin sollten wissenschaftliche Annotationen auf jeden Fall durch den Einsatz einer Datenkategorisierungsregistrierung im Sinne von ISOcat semantisch fundiert werden, während XStandoff als mögliches intermediäres Format zur Speicherung und Analyse multipler Annotation genutzt werden kann.

Welcome to advanced ODD! You're entering new territory with this. In anything comparable I have done, I have defined the ‚other language attribute' in my ODD. [...] Your desire to point to an externally-defined attribute needs some thought on how to express and implement it..." Der Beitrag kann unter der URL `http://listserv.brown.edu/archives/cgi-bin/wa?A2=ind1003&L=TEI-L&T=0&F=&S=&P=17759` eingesehen werden, zuletzt abgerufen am 19.04.2012.

Teil V

Anhang

Dokumentgrammatik für XStandoff

Die beiden folgenden Listings enthalten die vollständigen aktuellen Fassung der Dokumentgrammatik von XStandoff sowohl in der XML Schema-Version 1.0 als auch in 1.1 (jeweils mit Co-Constraints, im ersten Fall realisiert durch eingebettete Schematron-Assertions, im zweiten Fall durch entsprechende XSD-Konstrukte). Beide Dateien sind darüber hinaus unter `http://www.xstandoff.net` frei zum Download verfügbar.[2]

Listing 1: XStandoff-Dokumentgrammatik (XML Schema 1.0)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   xmlns:sch="http://purl.oclc.org/dsdl/schematron"
4   targetNamespace="http://www.xstandoff.net/2009/xstandoff/1.1"
5   xmlns:all="http://www.xstandoff.net/2009/all" elementFormDefault="qualified"
6   xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1">
7   <xs:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="xml.xsd"/>
8   <xs:include schemaLocation="log.xsd"/>
9   <xs:annotation>
10   <xs:documentation xml:lang="en">
11    <![CDATA[
12      XSD for the XStandoff format (XSF) base layer
13      Version: 1.2r
14      Date: 2012-01-03
15      Copyright (c) 2008-2012 Maik Stührenberg, Bielefeld University.
16      Contact: maik.stuehrenberg@uni-bielefeld.de
17      This program is free software: you can redistribute it and/or modify it under the terms
            of the GNU Lesser General Public License as published by the Free Software
          Foundation, either version 3 of the License, or (at your option) any later version.
18      This program is distributed in the hope that it will be useful, but WITHOUT ANY
          WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
          PARTICULAR PURPOSE. See the GNU GNU Lesser General Public License for more details.
19      You should have received a copy of the GNU Lesser General Public License along with
          this program (file 'lgpl-3.0.txt'). If not, see <http://www.gnu.org/licenses/>.
20      ]]>
21   </xs:documentation>
22  </xs:annotation>
23  <!-- Element declaration -->
24  <xs:element name="corpus">
25   <xs:annotation>
26    <xs:documentation xml:lang="en">The corpus element stores a whole corpus containing of
        resources (optional), meta data (optional) and corpusData entries (normally texts,
        other file types including video and audio files are supported as well). Usually, one
         would use the possibility to use one file containing the corpus root element which
        consists of several corpusDataRef elements (e.g. pointers to separate corpusData
        instances).</xs:documentation>
27   </xs:annotation>
28   <xs:complexType>
29    <xs:sequence>
30     <xs:choice>
```

[2] Die XML-Schemata sind unter der URL `http://www.xstandoff.net/2009/xstandoff/1.1/xsf.xsd` (Version 1.0) bzw. `http://www.xstandoff.net/2009/xstandoff/1.1/xsf_1.1.xsd` zu finden, zuletzt abgerufen am 19.04.2012.

```
31      <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
32      <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
33     </xs:choice>
34     <xs:element ref="xsf:resources" minOccurs="0" maxOccurs="1"/>
35     <xs:choice minOccurs="0" maxOccurs="unbounded">
36      <xs:element ref="xsf:corpusData"/>
37      <xs:element ref="xsf:corpusDataRef"/>
38     </xs:choice>
39    </xs:sequence>
40   </xs:complexType>
41  </xs:element>
42  <xs:element name="corpusData">
43   <xs:annotation>
44    <xs:documentation xml:lang="en">The corpusData element stores all information regarding a
          single corpus file. </xs:documentation>
45    <xs:appinfo>
46     <sch:title>Schematron primaryData constraint rule</sch:title>
47     <sch:ns prefix="base" uri="http://www.text-technology.de/sekimo"/>
48     <sch:pattern id="PositionValues">
49      <sch:rule context="xsf:corpusData">
50       <sch:assert test="count(xsf:primaryData) = 1 or count(xsf:primaryData) > 1 and not(
              xsf:primaryData[@role='master'])" >None of the provided primaryData elements is set
               as master file.</sch:assert>
51      </sch:rule>
52     </sch:pattern>
53    </xs:appinfo>
54   </xs:annotation>
55   <xs:complexType>
56    <xs:sequence>
57     <xs:choice>
58      <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
59      <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
60     </xs:choice>
61     <xs:element ref="xsf:primaryData" minOccurs="1" maxOccurs="unbounded"/>
62     <xs:element ref="xsf:segmentation" minOccurs="0" maxOccurs="1"/>
63     <xs:element ref="xsf:annotation" minOccurs="0" maxOccurs="1"/>
64    </xs:sequence>
65    <xs:attributeGroup ref="xsf:cd.attributes"/>
66    <xs:attribute ref="xsf:alt" use="optional"/>
67   </xs:complexType>
68  </xs:element>
69  <xs:element name="corpusDataRef">
70   <xs:annotation>
71    <xs:documentation xml:lang="en">The corpusDataRef element is used as a placeholder for a
          single corpusData entry saved externally.</xs:documentation>
72   </xs:annotation>
73   <xs:complexType>
74    <xs:complexContent>
75     <xs:extension base="xsf:RefType">
76      <xs:attributeGroup ref="xsf:cd.attributes"/>
77     </xs:extension>
78    </xs:complexContent>
79   </xs:complexType>
80  </xs:element>
81  <xs:element name="metaRef" type="xsf:RefType">
82   <xs:annotation>
83    <xs:documentation xml:lang="en">The metaRef element is used as a placeholder for external
          metadata.</xs:documentation>
84   </xs:annotation>
85  </xs:element>
86  <xs:element name="primaryData">
87   <xs:annotation>
```

```
88    <xs:documentation xml:lang="en">The primaryData element contains either one (or more)
          reference(s) to a normalized (in terms of whitespace) file serving as base data for
          the annotation, or -- when dealing with shorter texts -- the whole text as a string
          underneath the textualContent child element.</xs:documentation>
89   </xs:annotation>
90   <xs:complexType>
91    <xs:sequence>
92     <xs:choice>
93      <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
94      <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
95     </xs:choice>
96     <xs:choice>
97      <xs:element ref="xsf:textualContent" minOccurs="1" maxOccurs="1"/>
98      <xs:element ref="xsf:primaryDataRef" maxOccurs="unbounded"/>
99     </xs:choice>
100    <xs:element ref="xsf:checksum" minOccurs="0" maxOccurs="unbounded"/>
101   </xs:sequence>
102   <xs:attributeGroup ref="xsf:base.attributes"/>
103   <xs:attributeGroup ref="xsf:pd.attributes"/>
104   <xs:attribute ref="xml:id" use="optional"/>
105   <xs:attribute ref="xml:lang" use="optional"/>
106   <xs:attribute ref="xml:space" use="optional"/>
107  </xs:complexType>
108 </xs:element>
109 <xs:element name="primaryDataRef" type="xsf:RefType">
110  <xs:annotation>
111   <xs:documentation xml:lang="en">The primaryDataRef element is used to refer to one or
          externally stored representations of the primary data used during the annotation
          process.</xs:documentation>
112  </xs:annotation>
113 </xs:element>
114 <xs:element name="textualContent" type="xs:string">
115  <xs:annotation>
116   <xs:documentation xml:lang="en">The textualContent element stores the textual primary
          data.</xs:documentation>
117  </xs:annotation>
118 </xs:element>
119 <xs:element name="checksum">
120  <xs:annotation>
121   <xs:documentation xml:lang="en">The checksum element can be used to preserve integrity of
           the primary data when dealing with multiple annotation layers.</xs:documentation>
122  </xs:annotation>
123  <xs:complexType mixed="true">
124   <xs:attribute name="algorithm" type="xs:string" use="required">
125    <xs:annotation>
126     <xs:documentation xml:lang="en">It is required to supply the algorithm that was used to
           calculate the checksum.</xs:documentation>
127    </xs:annotation>
128   </xs:attribute>
129  </xs:complexType>
130 </xs:element>
131 <xs:element name="resources">
132  <xs:annotation>
133   <xs:documentation xml:lang="en">The resources element is a container element for the
          resources used for segmentation or annotation of the primary data. Resources can be
          included in an XSF file (via the resource element) or referred to by the resourceRef
          element.</xs:documentation>
134  </xs:annotation>
135  <xs:complexType>
136   <xs:sequence minOccurs="1" maxOccurs="unbounded">
137    <xs:choice minOccurs="0" maxOccurs="unbounded">
138     <xs:element ref="xsf:meta"/>
139     <xs:element ref="xsf:metaRef"/>
```

```
140     </xs:choice>
141     <xs:element ref="xsf:resource"/>
142    </xs:sequence>
143   </xs:complexType>
144  </xs:element>
145  <xs:element name="segmentation">
146   <xs:annotation>
147    <xs:documentation xml:lang="en">The segmentation element is a container element for the
         segmentation units (segment, one or more) that are used to identify text spans in the
          primary data.</xs:documentation>
148   </xs:annotation>
149   <xs:complexType>
150    <xs:sequence>
151     <xs:choice>
152      <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
153      <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
154     </xs:choice>
155     <xs:element ref="xsf:segment" minOccurs="1" maxOccurs="unbounded"/>
156    </xs:sequence>
157   </xs:complexType>
158  </xs:element>
159  <xs:element name="segment">
160   <xs:annotation>
161    <xs:documentation xml:lang="en">A segment identifies a (character or time) span over the
         primary data. The creator attribute may be used for referencing a resource that was
         used during segmentation, in addition, the alt attribute may be used to mark this
         segment as an alternative segmentation to other segments.</xs:documentation>
162    <xs:appinfo>
163     <sch:title>Schematron segment constraint rule</sch:title>
164     <sch:ns prefix="base" uri="http://www.text-technology.de/sekimo"/>
165     <sch:pattern id="PositionValues">
166      <sch:rule context="xsf:segment">
167       <sch:assert test="xs:integer(@end) >= xs:integer(@start)" diagnostics="checkPositions">
            End position of a segment must be higher than or equal to its start position.</
            sch:assert>
168       <sch:assert test="xs:integer(@start) >= xs:integer(//xsf:corpusData/xsf:primaryData/
            @start)" diagnostics="checkPositionsPrimaryDataStart">Start position of a segment
            must be higher than or equal to the start position of the complete primary data.</
            sch:assert>
169       <sch:assert test="xs:integer(@end) &lt;= xs:integer(//xsf:corpusData/xsf:primaryData/
            @end)" diagnostics="checkPositionsPrimaryDataEnd">End position of a segment must be
             lower than or equal to the position of the complete primary data.</sch:assert>
170      </sch:rule>
171     </sch:pattern>
172     <sch:diagnostics>
173      <sch:diagnostic id="checkPositions">DiagnosticsError: End position of segment <sch:value
           -of select="@xml:id"/> is lower than its start position. Start position: <sch:value-
           of select="@start"/>. End position: <sch:value-of select="@end"/>.</sch:diagnostic>
174      <sch:diagnostic id="checkPositionsPrimaryDataStart">DiagnosticsError: Start position of
           segment <sch:value-of select="@xml:id"/> is lower than start position of primary
           data. Start position of segment: <sch:value-of select="@start"/>. Start position of
            primary data: <sch:value-of select="//xsf:corpusData/xsf:primaryData/@start"/>.</
           sch:diagnostic>
175      <sch:diagnostic id="checkPositionsPrimaryDataEnd">DiagnosticsError: End position of
           segment <sch:value-of select="@xml:id"/> is greater than end position of primary
           data. End position of segment: <sch:value-of select="@end"/>. End position of
           primary data: <sch:value-of select="//xsf:corpusData/xsf:primaryData/@end"/>.</
           sch:diagnostic>
176     </sch:diagnostics>
177    </xs:appinfo>
178   </xs:annotation>
179   <xs:complexType>
180    <xs:choice>
```

```
    <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xsf:base.attributes"/>
   <xs:attributeGroup ref="xsf:seg.attributes"/>
   <xs:attribute ref="xsf:creator" use="optional"/>
   <xs:attribute ref="xsf:alt" use="optional"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="resource">
  <xs:annotation>
   <xs:documentation xml:lang="en">The resource element contains optional meta data and the
       resource used for segmentation or annotation.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element ref="xsf:meta"/>
     <xs:element ref="xsf:metaRef"/>
    </xs:choice>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element ref="xsf:resourceContent" minOccurs="1" maxOccurs="1"/>
     <xs:element ref="xsf:resourceRef"/>
    </xs:choice>
   </xs:sequence>
   <xs:attribute ref="xml:id" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="resourceRef" type="xsf:RefType">
  <xs:annotation>
   <xs:documentation xml:lang="en">The resourceRef element is used for referencing resources
        used in the annotation process that are stored externally.</xs:documentation>
  </xs:annotation>
 </xs:element>
 <xs:element name="resourceContent">
  <xs:annotation>
   <xs:documentation xml:lang="en">The resourceContent contains the content of a resource.</
       xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="annotation">
  <xs:annotation>
   <xs:documentation xml:lang="en">The annotation element contains annotation levels (and
       their respective layer(s)).</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:choice>
     <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:element ref="xsf:level" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element ref="xsf:log" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="level">
  <xs:annotation>
```

```
240     <xs:documentation xml:lang="en">The level element stores meta information and the actual
            annotation layer(s). A level refers to the conceptual model of information
            represented in markup.</xs:documentation>
241    </xs:annotation>
242    <xs:complexType>
243     <xs:sequence>
244      <xs:choice>
245       <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
246       <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
247      </xs:choice>
248      <xs:element ref="xsf:layer" minOccurs="1" maxOccurs="unbounded"/>
249      <xs:element ref="xsf:log" minOccurs="0"/>
250     </xs:sequence>
251     <xs:attribute ref="xml:id" use="optional"/>
252     <xs:attribute name="import" type="xs:IDREFS" use="optional">
253      <xs:annotation>
254       <xs:documentation xml:lang="en">The import attribute should be used to mark other
            annotations levels that have to be established before this level can be used (i.e.
            this annotation level uses information derived from other levels).</
            xs:documentation>
255      </xs:annotation>
256     </xs:attribute>
257     <xs:attribute ref="xsf:dcrCategory" use="optional"/>
258    </xs:complexType>
259   </xs:element>
260   <xs:element name="meta">
261    <xs:annotation>
262     <xs:documentation xml:lang="en">The meta element can be used to store meta information.
            Elements derived from other namespaces are allowed, e.g. OLAC metadata is recommended
             for textual primary data. Note that schema processing is set to lax, i.e. a meta
            information schema can be provided but is optional.</xs:documentation>
263    </xs:annotation>
264    <xs:complexType>
265     <xs:sequence>
266      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
267     </xs:sequence>
268    </xs:complexType>
269   </xs:element>
270   <xs:element name="layer">
271    <xs:annotation>
272     <xs:documentation xml:lang="en">In contrast to the level the layer element refers to the
            technical realisation (or XML serialisation) of markup. It contains the converted (i.
            e. standoff) representation of inline annotation layers and their respective elements
             and attributes. Note that schema processing is set to lax, i.e. an underlying schema
             (XSD) should be accessible, i.e., all layers should have an corresponding XSD
            associated, the only exception should be the all layer. The priority attribute can be
             used to prioritize annotation levels (in case of overlapping markup during an inline
             markup unification) -- the highest number correspond to the innermost markup. The
            creator attribute can be used to refer to resources that have been provided in an XSF
             file or to a user/agent who has made an annotation level. Keep in mind, that its
            type is NMTOKENS since resources belong to the corpus as a whole, not to a single
            corpusData file</xs:documentation>
273    </xs:annotation>
274    <xs:complexType>
275     <xs:sequence>
276      <xs:choice>
277       <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
278       <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
279      </xs:choice>
280      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
281      <xs:element ref="xsf:log" minOccurs="0"/>
282     </xs:sequence>
283     <xs:attribute ref="xml:id" use="optional"/>
```

```
    <xs:attribute name="priority" type="xs:int" default="0"/>
    <xs:attribute ref="xsf:creator" use="optional"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="inline">
   <xs:annotation>
    <xs:documentation xml:lang="en">The inline element is used as root element for the
        outcome of an XStandoff2Inline transformation.</xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:sequence>
     <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="milestone">
   <xs:annotation>
    <xs:documentation xml:lang="en">The milestone element is modelled after the TEI milestone
         element and should be used when an XStandoff instance should be serialized into an
        inline annotation format. The xsf:segment attribute can be used to refer to the
        segment that was used in the XStandoff representation, the charPos attribute depicts
        the character position in the primary data where the overlap happened.</
        xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:attribute ref="xsf:segment" use="optional"/>
    <xs:attribute name="n" use="optional"/>
    <xs:attribute name="charPos" use="required" type="xs:positiveInteger"/>
    <xs:attribute name="unit" use="required" type="xs:NMTOKENS"/>
    <xs:attribute name="mType">
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="start"/>
       <xs:enumeration value="end"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:attribute>
   </xs:complexType>
  </xs:element>
  <!-- Attribute group declaration -->
  <xs:attributeGroup name="cd.attributes">
   <xs:attribute ref="xml:id" use="required"/>
   <xs:attribute name="xsfVersion" type="xs:decimal" default="1.1"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="seg.attributes">
   <xs:attribute ref="xml:id" use="required"/>
   <xs:attribute name="primaryData" type="xs:IDREFS" use="optional"/>
   <xs:attribute name="type" use="optional">
    <xs:annotation>
     <xs:documentation xml:lang="en">The type attribute shall be used to specify the segment.
          Possible values are: 'empty' for empty segments depicting formerly empty inline
         elements, 'char' for (non-whitespace) character data of the primary data, 'ws' for
         whitespace character data, 'pun' for punctuation characters, 'dur' when using a
         duration (i.e. non-character based) segmentation, 'seg' for depicting a segment that
          is composed via other segments.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="empty"/>
      <xs:enumeration value="char"/>
      <xs:enumeration value="ws"/>
      <xs:enumeration value="pun"/>
      <xs:enumeration value="dur"/>
```

```
336       <xs:enumeration value="seg"/>
337      </xs:restriction>
338     </xs:simpleType>
339    </xs:attribute>
340    <xs:attribute name="content" type="xs:string" use="optional">
341     <xs:annotation>
342      <xs:documentation xml:lang="en">In the very rare case that you want to store the textual
           content of a segment, that is the primary data that is annotated by the annotation
          elements referring to this very segment, use the content attribute. The content can
          be saved in terms of literal characters or in terms of Unicode codepoints. See the
          unicodeNormalizationForm attribute in the latter case.</xs:documentation>
343     </xs:annotation>
344    </xs:attribute>
345    <xs:attribute name="unicodeNormalizationForm" use="optional">
346     <xs:annotation>
347      <xs:documentation xml:lang="en">Some special characters (such as accented characters)
          can be represented in Unicode character sets by unique code points (precomposed) or
          with combining characters (decomposed). Since XSF requires a unique representation,
          a normalized form of Unicode text shall be used if a Unicode character set is used.
          Four Unicode Standard normalization forms are available: Normalization Form D (NFD),
           Normalization Form KD (NFKD), Normalization Form C (NFC), and Normalization Form KC
           (NFKC). Roughly speaking, NFD and NFKD decompose characters where possible, while
          NFC and NFKC compose characters where possible. Cf. Chapter 2, 'General Structure',
          Chapter 3, 'Conformance' and the Annex #15, 'Unicode Normalization Forms,' of The
          Unicode Standard for details. This information may be used by software processing
          XStandoff instances.</xs:documentation>
348     </xs:annotation>
349     <xs:simpleType>
350      <xs:restriction base="xs:string">
351       <xs:enumeration value="NFD"/>
352       <xs:enumeration value="NFKD"/>
353       <xs:enumeration value="NFC"/>
354       <xs:enumeration value="NFKC"/>
355      </xs:restriction>
356     </xs:simpleType>
357    </xs:attribute>
358    <xs:attribute name="segments" type="xs:IDREFS" use="optional">
359     <xs:annotation>
360      <xs:documentation xml:lang="en">The segments attribute can be used together with the
          mode attribute to create a new segment via referring to already established segments.
          </xs:documentation>
361     </xs:annotation>
362    </xs:attribute>
363    <xs:attribute name="mode" default="continuous">
364     <xs:annotation>
365      <xs:documentation xml:lang="en">The mode attribute marks segments that are created by
          referring to already established segments (cf. the segments attribute) either as
          continuous or disjoint.</xs:documentation>
366     </xs:annotation>
367     <xs:simpleType>
368      <xs:restriction base="xs:string">
369       <xs:enumeration value="continuous"/>
370       <xs:enumeration value="disjoint"/>
371      </xs:restriction>
372     </xs:simpleType>
373    </xs:attribute>
374   </xs:attributeGroup>
375   <xs:attributeGroup name="base.attributes">
376    <xs:attribute name="start" type="xs:nonNegativeInteger" use="optional"/>
377    <xs:attribute name="end" type="xs:nonNegativeInteger" use="optional"/>
378   </xs:attributeGroup>
379   <xs:attributeGroup name="pd.attributes">
380    <xs:attribute name="role" default="master">
```

```
    <xs:annotation>
     <xs:documentation xml:lang="en">In case of multiple primary data files one primary data
          should be set as master using the role attribute. The master primary data file
          specifies the timeline/character stream that is used for creating segments.</
          xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="master"/>
      <xs:enumeration value="slave"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="unit" use="optional" default="chars">
    <xs:annotation>
     <xs:documentation xml:lang="en">The unit used for delimiting the segmentation. XSF does
          not use the built-in data type duration, since it lacks some of the units used here.
          </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="chars"/>
      <xs:enumeration value="bytes"/>
      <xs:enumeration value="microseconds"/>
      <xs:enumeration value="milliseconds"/>
      <xs:enumeration value="seconds"/>
      <xs:enumeration value="minutes"/>
      <xs:enumeration value="hours"/>
      <xs:enumeration value="days"/>
      <xs:enumeration value="weeks"/>
      <xs:enumeration value="months"/>
      <xs:enumeration value="years"/>
      <xs:enumeration value="palFrames"/>
      <xs:enumeration value="ntscFrames"/>
      <xs:enumeration value="other"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="offset" use="optional" type="xs:integer">
    <xs:annotation>
     <xs:documentation xml:lang="en">The offset attribute shall be used for specifying an
          offset (in the same unit that is used for delimiting segments) between different
          primaryData files.</xs:documentation>
    </xs:annotation>
   </xs:attribute>
  </xs:attributeGroup>
  <xs:attributeGroup name="ref.attributes">
   <xs:attribute name="uri" type="xs:anyURI" use="required">
    <xs:annotation>
     <xs:documentation xml:lang="en">The URI specifying the location of the referenced file.</
          xs:documentation>
    </xs:annotation>
   </xs:attribute>
   <xs:attribute name="uriType" use="optional" default="relative">
    <xs:annotation>
     <xs:documentation xml:lang="en">This attribute is used to differentiate between relative
           and absolute URIs.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="relative"/>
      <xs:enumeration value="absolute"/>
     </xs:restriction>
```

```
435     </xs:simpleType>
436    </xs:attribute>
437    <xs:attribute name="encoding" type="xs:string" use="optional" default="utf-8"/>
438    <xs:attribute name="unicodeNormalizationForm" use="optional">
439     <xs:annotation>
440      <xs:documentation xml:lang="en">Some special characters (such as accented characters)
           can be represented in Unicode character sets by unique code points (precomposed) or
           with combining characters (decomposed). Since XSF requires a unique representation,
           a normalized form of Unicode text shall be used if a Unicode character set is used.
           Four Unicode Standard normalization forms are available: Normalization Form D (NFD),
            Normalization Form KD (NFKD), Normalization Form C (NFC), and Normalization Form KC
            (NFKC). Roughly speaking, NFD and NFKD decompose characters where possible, while
           NFC and NFKC compose characters where possible. Cf. Chapter 2, 'General Structure',
           Chapter 3, 'Conformance' and the Annex #15, 'Unicode Normalization Forms,' of The
           Unicode Standard for details. These information may be used by software processing
           XStandoff instances.</xs:documentation>
441     </xs:annotation>
442     <xs:simpleType>
443      <xs:restriction base="xs:string">
444       <xs:enumeration value="NFD"/>
445       <xs:enumeration value="NFKD"/>
446       <xs:enumeration value="NFC"/>
447       <xs:enumeration value="NFKC"/>
448      </xs:restriction>
449     </xs:simpleType>
450    </xs:attribute>
451    <xs:attribute name="mimeType" use="optional">
452     <xs:annotation>
453      <xs:documentation xml:lang="en">Enumeration of mime-types that can be used for primary
           data files.</xs:documentation>
454     </xs:annotation>
455     <xs:simpleType>
456      <xs:restriction base="xs:string">
457       <xs:enumeration value="application/x-bytecode.python"/>
458       <xs:enumeration value="application/acad"/>
459       <xs:enumeration value="application/applefile"/>
460       <xs:enumeration value="application/astound"/>
461       <xs:enumeration value="application/arj"/>
462       <xs:enumeration value="application/base64"/>
463       <xs:enumeration value="application/binhex"/>
464       <xs:enumeration value="application/binhex4"/>
465       <xs:enumeration value="application/book"/>
466       <xs:enumeration value="application/cdf"/>
467       <xs:enumeration value="application/clariscad"/>
468       <xs:enumeration value="application/commonground"/>
469       <xs:enumeration value="application/drafting"/>
470       <xs:enumeration value="application/dsptype"/>
471       <xs:enumeration value="application/dxf"/>
472       <xs:enumeration value="application/envoy"/>
473       <xs:enumeration value="application/excel"/>
474       <xs:enumeration value="application/fractals"/>
475       <xs:enumeration value="application/freeloader"/>
476       <xs:enumeration value="application/futuresplash"/>
477       <xs:enumeration value="application/gnutar"/>
478       <xs:enumeration value="application/groupwise"/>
479       <xs:enumeration value="application/hlp"/>
480       <xs:enumeration value="application/hta"/>
481       <xs:enumeration value="application/i-deas"/>
482       <xs:enumeration value="application/iges"/>
483       <xs:enumeration value="application/iges"/>
484       <xs:enumeration value="application/inf"/>
485       <xs:enumeration value="application/java"/>
486       <xs:enumeration value="application/java-byte-code"/>
```

```
487     <xs:enumeration value="application/lha"/>
488     <xs:enumeration value="application/lzx"/>
489     <xs:enumeration value="application/mac-binary"/>
490     <xs:enumeration value="application/mac-binhex"/>
491     <xs:enumeration value="application/mac-binhex40"/>
492     <xs:enumeration value="application/mac-compactpro"/>
493     <xs:enumeration value="application/macbinary"/>
494     <xs:enumeration value="application/marc"/>
495     <xs:enumeration value="application/mbedlet"/>
496     <xs:enumeration value="application/mcad"/>
497     <xs:enumeration value="application/mime"/>
498     <xs:enumeration value="application/mspowerpoint"/>
499     <xs:enumeration value="application/msword"/>
500     <xs:enumeration value="application/mswrite"/>
501     <xs:enumeration value="application/netmc"/>
502     <xs:enumeration value="application/octet-stream"/>
503     <xs:enumeration value="application/oda"/>
504     <xs:enumeration value="application/pdf"/>
505     <xs:enumeration value="application/pkcs-12"/>
506     <xs:enumeration value="application/pkcs-crl"/>
507     <xs:enumeration value="application/pkcs10"/>
508     <xs:enumeration value="application/pkcs7-mime"/>
509     <xs:enumeration value="application/pkcs7-signature"/>
510     <xs:enumeration value="application/pkix-cert"/>
511     <xs:enumeration value="application/pkix-crl"/>
512     <xs:enumeration value="application/plain"/>
513     <xs:enumeration value="application/postscript"/>
514     <xs:enumeration value="application/powerpoint"/>
515     <xs:enumeration value="application/pro_eng"/>
516     <xs:enumeration value="application/ringing-tones"/>
517     <xs:enumeration value="application/rtf"/>
518     <xs:enumeration value="application/rtf"/>
519     <xs:enumeration value="application/sdp"/>
520     <xs:enumeration value="application/sea"/>
521     <xs:enumeration value="application/set"/>
522     <xs:enumeration value="application/sla"/>
523     <xs:enumeration value="application/smil"/>
524     <xs:enumeration value="application/solids"/>
525     <xs:enumeration value="application/sounder"/>
526     <xs:enumeration value="application/step"/>
527     <xs:enumeration value="application/streamingmedia"/>
528     <xs:enumeration value="application/toolbook"/>
529     <xs:enumeration value="application/vda"/>
530     <xs:enumeration value="application/vnd.fdf"/>
531     <xs:enumeration value="application/vnd.hp-hpgl"/>
532     <xs:enumeration value="application/vnd.hp-pcl"/>
533     <xs:enumeration value="application/vnd.ms-excel"/>
534     <xs:enumeration value="application/vnd.ms-pki.certstore"/>
535     <xs:enumeration value="application/vnd.ms-pki.pko"/>
536     <xs:enumeration value="application/vnd.ms-pki.seccat"/>
537     <xs:enumeration value="application/vnd.ms-pki.stl"/>
538     <xs:enumeration value="application/vnd.ms-powerpoint"/>
539     <xs:enumeration value="application/vnd.ms-project"/>
540     <xs:enumeration value="application/vnd.nokia.configuration-message"/>
541     <xs:enumeration value="application/vnd.nokia.ringing-tone"/>
542     <xs:enumeration value="application/vnd.rn-realmedia"/>
543     <xs:enumeration value="application/vnd.rn-realplayer"/>
544     <xs:enumeration value="application/vnd.wap.wmlc"/>
545     <xs:enumeration value="application/vnd.wap.wmlscriptc"/>
546     <xs:enumeration value="application/vnd.xara"/>
547     <xs:enumeration value="application/vocaltec-media-desc"/>
548     <xs:enumeration value="application/vocaltec-media-file"/>
549     <xs:enumeration value="application/wordperfect"/>
```

```
550     <xs:enumeration value="application/wordperfect6.0"/>
551     <xs:enumeration value="application/wordperfect6.1"/>
552     <xs:enumeration value="application/x-123"/>
553     <xs:enumeration value="application/x-aim"/>
554     <xs:enumeration value="application/x-authorware-bin"/>
555     <xs:enumeration value="application/x-authorware-map"/>
556     <xs:enumeration value="application/x-authorware-seg"/>
557     <xs:enumeration value="application/x-bcpio"/>
558     <xs:enumeration value="application/x-binary"/>
559     <xs:enumeration value="application/x-binhex40"/>
560     <xs:enumeration value="application/x-bsh"/>
561     <xs:enumeration value="application/x-bytecode.elisp (compiled elisp)"/>
562     <xs:enumeration value="application/x-bzip"/>
563     <xs:enumeration value="application/x-bzip2"/>
564     <xs:enumeration value="application/x-cdf"/>
565     <xs:enumeration value="application/x-cdlink"/>
566     <xs:enumeration value="application/x-chat"/>
567     <xs:enumeration value="application/x-chat"/>
568     <xs:enumeration value="application/x-cmu-raster"/>
569     <xs:enumeration value="application/x-cocoa"/>
570     <xs:enumeration value="application/x-compactpro"/>
571     <xs:enumeration value="application/x-compress"/>
572     <xs:enumeration value="application/x-compressed"/>
573     <xs:enumeration value="application/x-conference"/>
574     <xs:enumeration value="application/x-cpio"/>
575     <xs:enumeration value="application/x-cpt"/>
576     <xs:enumeration value="application/x-csh"/>
577     <xs:enumeration value="application/x-deepv"/>
578     <xs:enumeration value="application/x-director"/>
579     <xs:enumeration value="application/x-dvi"/>
580     <xs:enumeration value="application/x-elc"/>
581     <xs:enumeration value="application/x-envoy"/>
582     <xs:enumeration value="application/x-envoy"/>
583     <xs:enumeration value="application/x-esrehber"/>
584     <xs:enumeration value="application/x-excel"/>
585     <xs:enumeration value="application/x-frame"/>
586     <xs:enumeration value="application/x-freelance"/>
587     <xs:enumeration value="application/x-gsp"/>
588     <xs:enumeration value="application/x-gss"/>
589     <xs:enumeration value="application/x-gtar"/>
590     <xs:enumeration value="application/x-gzip"/>
591     <xs:enumeration value="application/x-hdf"/>
592     <xs:enumeration value="application/x-helpfile"/>
593     <xs:enumeration value="application/x-httpd-imap"/>
594     <xs:enumeration value="application/x-ima"/>
595     <xs:enumeration value="application/x-internett-signup"/>
596     <xs:enumeration value="application/x-inventor"/>
597     <xs:enumeration value="application/x-ip2"/>
598     <xs:enumeration value="application/x-java-class"/>
599     <xs:enumeration value="application/x-java-commerce"/>
600     <xs:enumeration value="application/x-javascript"/>
601     <xs:enumeration value="application/x-koan"/>
602     <xs:enumeration value="application/x-ksh"/>
603     <xs:enumeration value="application/x-latex"/>
604     <xs:enumeration value="application/x-lha"/>
605     <xs:enumeration value="application/x-lisp"/>
606     <xs:enumeration value="application/x-livescreen"/>
607     <xs:enumeration value="application/x-lotus"/>
608     <xs:enumeration value="application/x-lotusscreencam"/>
609     <xs:enumeration value="application/x-lzh"/>
610     <xs:enumeration value="application/x-lzx"/>
611     <xs:enumeration value="application/x-mac-binhex40"/>
612     <xs:enumeration value="application/x-macbinary"/>
```

```
    <xs:enumeration value="application/x-magic-cap-package-1.0"/>
    <xs:enumeration value="application/x-mathcad"/>
    <xs:enumeration value="application/x-meme"/>
    <xs:enumeration value="application/x-midi"/>
    <xs:enumeration value="application/x-mif"/>
    <xs:enumeration value="application/x-mix-transfer"/>
    <xs:enumeration value="application/x-mplayer2"/>
    <xs:enumeration value="application/x-msexcel"/>
    <xs:enumeration value="application/x-mspowerpoint"/>
    <xs:enumeration value="application/x-navi-animation"/>
    <xs:enumeration value="application/x-navidoc"/>
    <xs:enumeration value="application/x-navimap"/>
    <xs:enumeration value="application/x-navistyle"/>
    <xs:enumeration value="application/x-netcdf"/>
    <xs:enumeration value="application/x-netcdf"/>
    <xs:enumeration value="application/x-newton-compatible-pkg"/>
    <xs:enumeration value="application/x-nokia-9000-communicator-add-on-software"/>
    <xs:enumeration value="application/x-omc"/>
    <xs:enumeration value="application/x-omcdatamaker"/>
    <xs:enumeration value="application/x-omcregerator"/>
    <xs:enumeration value="application/x-pagemaker"/>
    <xs:enumeration value="application/x-pcl"/>
    <xs:enumeration value="application/x-pixclscript"/>
    <xs:enumeration value="application/x-pkcs10"/>
    <xs:enumeration value="application/x-pkcs12"/>
    <xs:enumeration value="application/x-pkcs7-certificates"/>
    <xs:enumeration value="application/x-pkcs7-certreqresp"/>
    <xs:enumeration value="application/x-pkcs7-mime"/>
    <xs:enumeration value="application/x-pkcs7-signature"/>
    <xs:enumeration value="application/x-pointplus"/>
    <xs:enumeration value="application/x-portable-anymap"/>
    <xs:enumeration value="application/x-project"/>
    <xs:enumeration value="application/x-qpro"/>
    <xs:enumeration value="application/x-rtf"/>
    <xs:enumeration value="application/x-sdp"/>
    <xs:enumeration value="application/x-sea"/>
    <xs:enumeration value="application/x-seelogo"/>
    <xs:enumeration value="application/x-sh"/>
    <xs:enumeration value="application/x-shar"/>
    <xs:enumeration value="application/x-shockwave-flash"/>
    <xs:enumeration value="application/x-sit"/>
    <xs:enumeration value="application/x-sprite"/>
    <xs:enumeration value="application/x-stuffit"/>
    <xs:enumeration value="application/x-sv4cpio"/>
    <xs:enumeration value="application/x-sv4crc"/>
    <xs:enumeration value="application/x-tar"/>
    <xs:enumeration value="application/x-tbook"/>
    <xs:enumeration value="application/x-tcl"/>
    <xs:enumeration value="application/x-tex"/>
    <xs:enumeration value="application/x-texinfo"/>
    <xs:enumeration value="application/x-troff"/>
    <xs:enumeration value="application/x-troff-man"/>
    <xs:enumeration value="application/x-troff-me"/>
    <xs:enumeration value="application/x-troff-ms"/>
    <xs:enumeration value="application/x-troff-msvideo"/>
    <xs:enumeration value="application/x-ustar"/>
    <xs:enumeration value="application/x-visio"/>
    <xs:enumeration value="application/x-vnd.audioexplosion.mzz"/>
    <xs:enumeration value="application/x-vnd.ls-xpix"/>
    <xs:enumeration value="application/x-vrml"/>
    <xs:enumeration value="application/x-wais-source"/>
    <xs:enumeration value="application/x-winhelp"/>
    <xs:enumeration value="application/x-wintalk"/>
```

```
676     <xs:enumeration value="application/x-world"/>
677     <xs:enumeration value="application/x-wpwin"/>
678     <xs:enumeration value="application/x-wri"/>
679     <xs:enumeration value="application/x-x509-ca-cert"/>
680     <xs:enumeration value="application/x-x509-user-cert"/>
681     <xs:enumeration value="application/x-zip-compressed"/>
682     <xs:enumeration value="application/xml"/>
683     <xs:enumeration value="application/zip"/>
684     <xs:enumeration value="audio/aiff"/>
685     <xs:enumeration value="audio/basic"/>
686     <xs:enumeration value="audio/it"/>
687     <xs:enumeration value="audio/make"/>
688     <xs:enumeration value="audio/make.my.funk"/>
689     <xs:enumeration value="audio/mid"/>
690     <xs:enumeration value="audio/midi"/>
691     <xs:enumeration value="audio/mod"/>
692     <xs:enumeration value="audio/mpeg"/>
693     <xs:enumeration value="audio/mpeg3"/>
694     <xs:enumeration value="audio/nspaudio"/>
695     <xs:enumeration value="audio/s3m"/>
696     <xs:enumeration value="audio/tsp-audio"/>
697     <xs:enumeration value="audio/tsplayer"/>
698     <xs:enumeration value="audio/vnd.qcelp"/>
699     <xs:enumeration value="audio/voc"/>
700     <xs:enumeration value="audio/voxware"/>
701     <xs:enumeration value="audio/wav"/>
702     <xs:enumeration value="audio/x-adpcm"/>
703     <xs:enumeration value="audio/x-aiff"/>
704     <xs:enumeration value="audio/x-au"/>
705     <xs:enumeration value="audio/x-gsm"/>
706     <xs:enumeration value="audio/x-jam"/>
707     <xs:enumeration value="audio/x-liveaudio"/>
708     <xs:enumeration value="audio/x-mid"/>
709     <xs:enumeration value="audio/x-midi"/>
710     <xs:enumeration value="audio/x-mod"/>
711     <xs:enumeration value="audio/x-mpeg"/>
712     <xs:enumeration value="audio/x-mpeg-3"/>
713     <xs:enumeration value="audio/x-mpequrl"/>
714     <xs:enumeration value="audio/x-nspaudio"/>
715     <xs:enumeration value="audio/x-pn-realaudio"/>
716     <xs:enumeration value="audio/x-pn-realaudio-plugin"/>
717     <xs:enumeration value="audio/x-psid"/>
718     <xs:enumeration value="audio/x-realaudio"/>
719     <xs:enumeration value="audio/x-twinvq"/>
720     <xs:enumeration value="audio/x-twinvq-plugin"/>
721     <xs:enumeration value="audio/x-vnd.audioexplosion.mjuicemediafile"/>
722     <xs:enumeration value="audio/x-voc"/>
723     <xs:enumeration value="audio/x-wav"/>
724     <xs:enumeration value="audio/xm"/>
725     <xs:enumeration value="chemical/x-pdb"/>
726     <xs:enumeration value="i-world/i-vrml"/>
727     <xs:enumeration value="image/bmp"/>
728     <xs:enumeration value="image/cmu-raster"/>
729     <xs:enumeration value="image/fif"/>
730     <xs:enumeration value="image/florian"/>
731     <xs:enumeration value="image/g3fax"/>
732     <xs:enumeration value="image/gif"/>
733     <xs:enumeration value="image/ief"/>
734     <xs:enumeration value="image/jpeg"/>
735     <xs:enumeration value="image/jutvision"/>
736     <xs:enumeration value="image/naplps"/>
737     <xs:enumeration value="image/pict"/>
738     <xs:enumeration value="image/pjpeg"/>
```

```
    <xs:enumeration value="image/png"/>
    <xs:enumeration value="image/tiff"/>
    <xs:enumeration value="image/vasa"/>
    <xs:enumeration value="image/vnd.dwg"/>
    <xs:enumeration value="image/vnd.fpx"/>
    <xs:enumeration value="image/vnd.net-fpx"/>
    <xs:enumeration value="image/vnd.rn-realflash"/>
    <xs:enumeration value="image/vnd.rn-realpix"/>
    <xs:enumeration value="image/vnd.wap.wbmp"/>
    <xs:enumeration value="image/vnd.xiff"/>
    <xs:enumeration value="image/x-cmu-raster"/>
    <xs:enumeration value="image/x-dwg"/>
    <xs:enumeration value="image/x-icon"/>
    <xs:enumeration value="image/x-jg"/>
    <xs:enumeration value="image/x-jps"/>
    <xs:enumeration value="image/x-niff"/>
    <xs:enumeration value="image/x-niff"/>
    <xs:enumeration value="image/x-pcx"/>
    <xs:enumeration value="image/x-pict"/>
    <xs:enumeration value="image/x-portable-anymap"/>
    <xs:enumeration value="image/x-portable-bitmap"/>
    <xs:enumeration value="image/x-portable-greymap"/>
    <xs:enumeration value="image/x-portable-pixmap"/>
    <xs:enumeration value="image/x-quicktime"/>
    <xs:enumeration value="image/x-rgb"/>
    <xs:enumeration value="image/x-tiff"/>
    <xs:enumeration value="image/x-windows-bmp"/>
    <xs:enumeration value="image/x-xbitmap"/>
    <xs:enumeration value="image/x-xbm"/>
    <xs:enumeration value="image/x-xpixmap"/>
    <xs:enumeration value="image/x-xwd"/>
    <xs:enumeration value="image/x-xwindowdump"/>
    <xs:enumeration value="image/xbm"/>
    <xs:enumeration value="image/xpm"/>
    <xs:enumeration value="message/rfc822"/>
    <xs:enumeration value="model/iges"/>
    <xs:enumeration value="model/vnd.dwf"/>
    <xs:enumeration value="model/vrml"/>
    <xs:enumeration value="model/x-pov"/>
    <xs:enumeration value="multipart/x-gzip"/>
    <xs:enumeration value="multipart/x-ustar"/>
    <xs:enumeration value="multipart/x-zip"/>
    <xs:enumeration value="music/crescendo"/>
    <xs:enumeration value="music/x-karaoke"/>
    <xs:enumeration value="paleovu/x-pv"/>
    <xs:enumeration value="text/asp"/>
    <xs:enumeration value="text/css"/>
    <xs:enumeration value="text/html"/>
    <xs:enumeration value="text/mcf"/>
    <xs:enumeration value="text/pascal"/>
    <xs:enumeration value="text/plain"/>
    <xs:enumeration value="text/richtext"/>
    <xs:enumeration value="text/scriplet"/>
    <xs:enumeration value="text/sgml"/>
    <xs:enumeration value="text/tab-separated-values"/>
    <xs:enumeration value="text/uri-list"/>
    <xs:enumeration value="text/vnd.abc"/>
    <xs:enumeration value="text/vnd.fmi.flexstor"/>
    <xs:enumeration value="text/vnd.rn-realtext"/>
    <xs:enumeration value="text/vnd.wap.wml"/>
    <xs:enumeration value="text/vnd.wap.wmlscript"/>
    <xs:enumeration value="text/webviewhtml"/>
    <xs:enumeration value="text/x-asm"/>
```

```
802     <xs:enumeration value="text/x-audiosoft-intra"/>
803     <xs:enumeration value="text/x-c"/>
804     <xs:enumeration value="text/x-component"/>
805     <xs:enumeration value="text/x-fortran"/>
806     <xs:enumeration value="text/x-h"/>
807     <xs:enumeration value="text/x-java-source"/>
808     <xs:enumeration value="text/x-la-asf"/>
809     <xs:enumeration value="text/x-m"/>
810     <xs:enumeration value="text/x-pascal"/>
811     <xs:enumeration value="text/x-script"/>
812     <xs:enumeration value="text/x-script.csh"/>
813     <xs:enumeration value="text/x-script.elisp"/>
814     <xs:enumeration value="text/x-script.guile"/>
815     <xs:enumeration value="text/x-script.ksh"/>
816     <xs:enumeration value="text/x-script.lisp"/>
817     <xs:enumeration value="text/x-script.perl"/>
818     <xs:enumeration value="text/x-script.perl-module"/>
819     <xs:enumeration value="text/x-script.phyton"/>
820     <xs:enumeration value="text/x-script.rexx"/>
821     <xs:enumeration value="text/x-script.scheme"/>
822     <xs:enumeration value="text/x-script.sh"/>
823     <xs:enumeration value="text/x-script.tcl"/>
824     <xs:enumeration value="text/x-script.tcsh"/>
825     <xs:enumeration value="text/x-script.zsh"/>
826     <xs:enumeration value="text/x-server-parsed-html"/>
827     <xs:enumeration value="text/x-setext"/>
828     <xs:enumeration value="text/x-sgml"/>
829     <xs:enumeration value="text/x-speech"/>
830     <xs:enumeration value="text/x-uil"/>
831     <xs:enumeration value="text/x-uuencode"/>
832     <xs:enumeration value="text/x-vcalendar"/>
833     <xs:enumeration value="text/xml"/>
834     <xs:enumeration value="video/animaflex"/>
835     <xs:enumeration value="video/avi"/>
836     <xs:enumeration value="video/avs-video"/>
837     <xs:enumeration value="video/dl"/>
838     <xs:enumeration value="video/fli"/>
839     <xs:enumeration value="video/gl"/>
840     <xs:enumeration value="video/mpeg"/>
841     <xs:enumeration value="video/msvideo"/>
842     <xs:enumeration value="video/quicktime"/>
843     <xs:enumeration value="video/vdo"/>
844     <xs:enumeration value="video/vivo"/>
845     <xs:enumeration value="video/vnd.rn-realvideo"/>
846     <xs:enumeration value="video/vnd.vivo"/>
847     <xs:enumeration value="video/vosaic"/>
848     <xs:enumeration value="video/x-amt-demorun"/>
849     <xs:enumeration value="video/x-amt-showrun"/>
850     <xs:enumeration value="video/x-atomic3d-feature"/>
851     <xs:enumeration value="video/x-dl"/>
852     <xs:enumeration value="video/x-dv"/>
853     <xs:enumeration value="video/x-dv"/>
854     <xs:enumeration value="video/x-fli"/>
855     <xs:enumeration value="video/x-gl"/>
856     <xs:enumeration value="video/x-isvideo"/>
857     <xs:enumeration value="video/x-motion-jpeg"/>
858     <xs:enumeration value="video/x-mpeg"/>
859     <xs:enumeration value="video/x-mpeq2a"/>
860     <xs:enumeration value="video/x-ms-asf"/>
861     <xs:enumeration value="video/x-ms-asf-plugin"/>
862     <xs:enumeration value="video/x-msvideo"/>
863     <xs:enumeration value="video/x-qtc"/>
864     <xs:enumeration value="video/x-scm"/>
```

```
865       <xs:enumeration value="video/x-sgi-movie"/>
866       <xs:enumeration value="windows/metafile"/>
867       <xs:enumeration value="www/mime"/>
868       <xs:enumeration value="x-conference/x-cooltalk"/>
869       <xs:enumeration value="x-music/x-midi"/>
870       <xs:enumeration value="x-world/x-3dmf"/>
871       <xs:enumeration value="x-world/x-svr"/>
872       <xs:enumeration value="x-world/x-vrml"/>
873       <xs:enumeration value="x-world/x-vrt"/>
874       <xs:enumeration value="xgl/drawing"/>
875       <xs:enumeration value="xgl/movie"/>
876       <xs:enumeration value="other"/>
877     </xs:restriction>
878    </xs:simpleType>
879   </xs:attribute>
880  </xs:attributeGroup>
881  <xs:attributeGroup name="imported">
882   <xs:annotation>
883    <xs:documentation xml:lang="en">This attribute group should be imported into the schema
          files of the respective layers. It consists of the obligatory segment attribute and
          the optional dcrCategory attribute.</xs:documentation>
884   </xs:annotation>
885   <xs:attribute ref="xsf:segment"/>
886   <xs:attribute ref="xsf:dcrCategory" use="optional"/>
887  </xs:attributeGroup>
888  <!-- Attribute declaration -->
889  <xs:attribute name="segment" type="xs:IDREFS"/>
890  <xs:attribute name="dcrCategory" type="xs:anyURI">
891   <xs:annotation>
892    <xs:documentation xml:lang="en">Use this attribute for providing a link to a ISO Data
          Category Registry category (cf. ISO 12620 for details).</xs:documentation>
893   </xs:annotation>
894  </xs:attribute>
895  <xs:attribute name="previousVersionLogEntry" type="xs:IDREF"/>
896  <xs:attribute name="creator" type="xs:NMTOKENS">
897   <xs:annotation>
898    <xs:documentation xml:lang="en">The creator attribute may be used for referring to (non-)
           human agent (e.g. a resource) that was involved in creating an item.</
          xs:documentation>
899   </xs:annotation>
900  </xs:attribute>
901  <xs:attribute name="affectedItem" type="xs:IDREF"/>
902  <xs:attribute name="alt" type="xs:IDREFS">
903   <xs:annotation>
904    <xs:documentation xml:lang="en">The alt attribute may be used for referring to an
          alternate version, neither in terms of isVersionOf, isBasedOn, isFormatOf or similar
          relations. The type of relation should be specified in the meta data entry of the
          respective element bearing the alt attribute.</xs:documentation>
905   </xs:annotation>
906  </xs:attribute>
907  <!-- complexType declaration -->
908  <xs:complexType name="RefType">
909   <xs:attributeGroup ref="xsf:ref.attributes"/>
910  </xs:complexType>
911 </xs:schema>
```

Listing 2: XStandoff-Dokumentgrammatik (XML Schema 1.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:all="http://www.xstandoff.net/2009/all" elementFormDefault="qualified"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1">
 <xs:import namespace="http://www.w3.org/XML/1998/namespace"
  schemaLocation="http://www.w3.org/2001/xml.xsd"/>
 <xs:include schemaLocation="log_1.1.xsd"/>
 <xs:annotation>
  <xs:documentation xml:lang="en">
   <![CDATA[
     XSD for the XStandoff format (XSF) base layer
     Version: 1.2u (XSD 1.1)
     Date: 2012-01-03
     Copyright (c) 2008-2012 Maik Stührenberg, Bielefeld University.
     Contact: maik.stuehrenberg@uni-bielefeld.de
     This program is free software: you can redistribute it and/or modify it under the terms
          of the GNU Lesser General Public License as published by the Free Software
         Foundation, either version 3 of the License, or (at your option) any later version.
     This program is distributed in the hope that it will be useful, but WITHOUT ANY
         WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
         PARTICULAR PURPOSE. See the GNU GNU Lesser General Public License for more details.
     You should have received a copy of the GNU Lesser General Public License along with
         this program (file 'lgpl-3.0.txt'). If not, see <http://www.gnu.org/licenses/>.
     ]]>
  </xs:documentation>
 </xs:annotation>
 <!-- Element declaration -->
 <xs:element name="corpus">
  <xs:annotation>
   <xs:documentation xml:lang="en">The corpus element stores a whole corpus containing of
       resources (optional), meta data (optional) and corpusData entries (normally texts,
       other file types including video and audio files are supported as well). Usually, one
        would use the possibility to use one file containing the corpus root element which
       consists of several corpusDataRef elements (e.g. pointers to separate corpusData
       instances).</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:choice>
     <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
    <xs:element ref="xsf:resources" minOccurs="0" maxOccurs="1"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element ref="xsf:corpusData"/>
     <xs:element ref="xsf:corpusDataRef"/>
    </xs:choice>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="corpusData">
  <xs:annotation>
   <xs:documentation xml:lang="en">The corpusData element stores all information regarding a
        single corpus file. </xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:choice>
     <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
```

```
52      <xs:element ref="xsf:primaryData" minOccurs="1" maxOccurs="unbounded"/>
53      <xs:element ref="xsf:segmentation" minOccurs="0" maxOccurs="1"/>
54      <xs:element ref="xsf:annotation" minOccurs="0" maxOccurs="1"/>
55     </xs:sequence>
56     <xs:attributeGroup ref="xsf:cd.attributes"/>
57     <xs:attribute ref="xsf:alt" use="optional"/>
58     <xs:assert
59      test="count(xsf:primaryData) = 1 or count(xsf:primaryData) > 1 and exactly-one(
            xsf:primaryData[@role='master'])">
60      <xs:annotation>
61       <xs:appinfo>There may either only be one primaryData element or exactly one primaryData
             element that has a role attribute containing the value 'master'.</xs:appinfo>
62       <xs:documentation xml:lang="en">Display the above error message if this assertion fails.
             </xs:documentation>
63      </xs:annotation>
64     </xs:assert>
65     <xs:assert
66      test="empty(for $p in xsf:primaryData[role='master']/@start, $s in xsf:segmentation/
            xsf:segment[$p le @start] return $s)">
67      <xs:annotation>
68       <xs:appinfo>No segment should have a start position that is lower than the start
             position of the master primaryData element.</xs:appinfo>
69       <xs:documentation xml:lang="en">Display the above error message if this assertion fails.
             </xs:documentation>
70      </xs:annotation>
71     </xs:assert>
72     <xs:assert
73      test="empty(for $p in xsf:primaryData[role='master']/@end, $s in xsf:segmentation/
            xsf:segment[$p ge @end] return $s)">
74      <xs:annotation>
75       <xs:appinfo>No segment should have an end position that is greater than the end position
              of the master primaryData element.</xs:appinfo>
76       <xs:documentation xml:lang="en">Display the above error message if this assertion fails.
             </xs:documentation>
77      </xs:annotation>
78     </xs:assert>
79    </xs:complexType>
80   </xs:element>
81   <xs:element name="corpusDataRef">
82    <xs:annotation>
83     <xs:documentation xml:lang="en">The corpusDataRef element is used as a placeholder for a
           single corpusData entry saved externally.</xs:documentation>
84    </xs:annotation>
85    <xs:complexType>
86     <xs:complexContent>
87      <xs:extension base="xsf:RefType">
88       <xs:attributeGroup ref="xsf:cd.attributes"/>
89      </xs:extension>
90     </xs:complexContent>
91    </xs:complexType>
92   </xs:element>
93   <xs:element name="metaRef" type="xsf:RefType">
94    <xs:annotation>
95     <xs:documentation xml:lang="en">The metaRef element is used as a placeholder for external
            metadata.</xs:documentation>
96    </xs:annotation>
97   </xs:element>
98   <xs:element name="primaryData">
99    <xs:annotation>
100    <xs:documentation xml:lang="en">The primaryData element contains either one (or more)
           reference(s) to a normalized (in terms of whitespace) file serving as base data for
           the annotation, or -- when dealing with shorter texts -- the whole text as a string
           underneath the textualContent child element.</xs:documentation>
```

```
101    </xs:annotation>
102    <xs:complexType>
103     <xs:sequence>
104      <xs:choice>
105       <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
106       <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
107      </xs:choice>
108      <xs:choice>
109       <xs:element ref="xsf:textualContent" minOccurs="1" maxOccurs="1"/>
110       <xs:element ref="xsf:primaryDataRef" maxOccurs="unbounded"/>
111      </xs:choice>
112      <xs:element ref="xsf:checksum" minOccurs="0" maxOccurs="unbounded"/>
113     </xs:sequence>
114     <xs:attributeGroup ref="xsf:base.attributes"/>
115     <xs:attributeGroup ref="xsf:pd.attributes"/>
116     <xs:attribute ref="xml:id" use="optional"/>
117     <xs:attribute ref="xml:lang" use="optional"/>
118     <xs:attribute ref="xml:space" use="optional"/>
119     <xs:assert test="xs:integer(@end) ge xs:integer(@start)">
120      <xs:annotation>
121       <xs:appinfo>The value of the start attribute must be less than or equal to the end
            attribute's value.</xs:appinfo>
122       <xs:documentation xml:lang="en">Display the above error message if this assertion fails
            .</xs:documentation>
123      </xs:annotation>
124     </xs:assert>
125    </xs:complexType>
126   </xs:element>
127   <xs:element name="primaryDataRef" type="xsf:RefType">
128    <xs:annotation>
129     <xs:documentation xml:lang="en">The primaryDataRef element is used to refer to one or
          externally stored representations of the primary data used during the annotation
          process.</xs:documentation>
130    </xs:annotation>
131   </xs:element>
132   <xs:element name="textualContent" type="xs:string">
133    <xs:annotation>
134     <xs:documentation xml:lang="en">The textualContent element stores the textual primary
          data.</xs:documentation>
135    </xs:annotation>
136   </xs:element>
137   <xs:element name="checksum">
138    <xs:annotation>
139     <xs:documentation xml:lang="en">The checksum element can be used to preserve integrity of
           the primary data when dealing with multiple annotation layers.</xs:documentation>
140    </xs:annotation>
141    <xs:complexType mixed="true">
142     <xs:attribute name="algorithm" type="xs:string" use="required">
143      <xs:annotation>
144       <xs:documentation xml:lang="en">It is required to supply the algorithm that was used to
            calculate the checksum.</xs:documentation>
145      </xs:annotation>
146     </xs:attribute>
147    </xs:complexType>
148   </xs:element>
149   <xs:element name="resources">
150    <xs:annotation>
151     <xs:documentation xml:lang="en">The resources element is a container element for the
          resources used for segmentation or annotation of the primary data. Resources can be
          included in an XSF file (via the resource element) or referred to by the resourceRef
          element.</xs:documentation>
152    </xs:annotation>
153    <xs:complexType>
```

```
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="xsf:meta"/>
    <xs:element ref="xsf:metaRef"/>
   </xs:choice>
   <xs:element ref="xsf:resource"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="segmentation">
 <xs:annotation>
  <xs:documentation xml:lang="en">The segmentation element is a container element for the
      segmentation units (segment, one or more) that are used to identify text spans in the
       primary data.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:sequence>
   <xs:choice>
    <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:element ref="xsf:segment" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="segment">
 <xs:annotation>
  <xs:documentation xml:lang="en">A segment identifies a (character or time) span over the
      primary data. The creator attribute may be used for referencing a resource that was
      used during segmentation, in addition, the alt attribute may be used to mark this
      segment as an alternative segmentation to other segments.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:choice>
   <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
  </xs:choice>
  <xs:attributeGroup ref="xsf:base.attributes"/>
  <xs:attributeGroup ref="xsf:seg.attributes"/>
  <xs:attribute ref="xsf:creator" use="optional"/>
  <xs:attribute ref="xsf:alt" use="optional"/>
  <xs:assert test="xs:integer(@end) ge xs:integer(@start)">
   <xs:annotation>
    <xs:appinfo>The value of the start attribute must be less than or equal to the end
        attribute's value.</xs:appinfo>
    <xs:documentation xml:lang="en">Display the above error message if this assertion fails.
        </xs:documentation>
   </xs:annotation>
  </xs:assert>
 </xs:complexType>
</xs:element>
<xs:element name="resource">
 <xs:annotation>
  <xs:documentation xml:lang="en">The resource element contains optional meta data and the
      resource used for segmentation or annotation.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:sequence>
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="xsf:meta"/>
    <xs:element ref="xsf:metaRef"/>
   </xs:choice>
   <xs:choice minOccurs="0" maxOccurs="unbounded">
```

```
209      <xs:element ref="xsf:resourceContent" minOccurs="1" maxOccurs="1"/>
210      <xs:element ref="xsf:resourceRef"/>
211     </xs:choice>
212    </xs:sequence>
213    <xs:attribute ref="xml:id" use="required"/>
214   </xs:complexType>
215  </xs:element>
216  <xs:element name="resourceRef" type="xsf:RefType">
217   <xs:annotation>
218    <xs:documentation xml:lang="en">The resourceRef element is used for referencing resources
          used in the annotation process that are stored externally.</xs:documentation>
219   </xs:annotation>
220  </xs:element>
221  <xs:element name="resourceContent">
222   <xs:annotation>
223    <xs:documentation xml:lang="en">The resourceContent contains the content of a resource.</
         xs:documentation>
224   </xs:annotation>
225   <xs:complexType>
226    <xs:sequence>
227     <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
228    </xs:sequence>
229   </xs:complexType>
230  </xs:element>
231  <xs:element name="annotation">
232   <xs:annotation>
233    <xs:documentation xml:lang="en">The annotation element contains annotation levels (and
         their respective layer(s)).</xs:documentation>
234   </xs:annotation>
235   <xs:complexType>
236    <xs:sequence>
237     <xs:choice>
238      <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
239      <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
240     </xs:choice>
241     <xs:element ref="xsf:level" minOccurs="1" maxOccurs="unbounded"/>
242     <xs:element ref="xsf:log" minOccurs="0"/>
243    </xs:sequence>
244   </xs:complexType>
245  </xs:element>
246  <xs:element name="level">
247   <xs:annotation>
248    <xs:documentation xml:lang="en">The level element stores meta information and the actual
         annotation layer(s). A level refers to the conceptual model of information
         represented in markup.</xs:documentation>
249   </xs:annotation>
250   <xs:complexType>
251    <xs:sequence>
252     <xs:choice>
253      <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
254      <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
255     </xs:choice>
256     <xs:element ref="xsf:layer" minOccurs="1" maxOccurs="unbounded"/>
257     <xs:element ref="xsf:log" minOccurs="0"/>
258    </xs:sequence>
259    <xs:attribute ref="xml:id" use="optional"/>
260    <xs:attribute name="import" type="xs:IDREFS" use="optional">
261     <xs:annotation>
262      <xs:documentation xml:lang="en">The import attribute should be used to mark other
           annotations levels that have to be established before this level can be used (i.e.
           this annotation level uses information derived from other levels).</
           xs:documentation>
263     </xs:annotation>
```

```
264     </xs:attribute>
265     <xs:attribute ref="xsf:dcrCategory" use="optional"/>
266    </xs:complexType>
267   </xs:element>
268   <xs:element name="meta">
269    <xs:annotation>
270     <xs:documentation xml:lang="en">The meta element can be used to store meta information.
          Elements derived from other namespaces are allowed, e.g. OLAC metadata is recommended
           for textual primary data. Note that schema processing is set to lax, i.e. a meta
          information schema can be provided but is optional.</xs:documentation>
271    </xs:annotation>
272    <xs:complexType>
273     <xs:sequence>
274      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
275     </xs:sequence>
276    </xs:complexType>
277   </xs:element>
278   <xs:element name="layer">
279    <xs:annotation>
280     <xs:documentation xml:lang="en">In contrast to the level the layer element refers to the
          technical realisation (or XML serialisation) of markup. It contains the converted (i.
          e. standoff) representation of inline annotation layers and their respective elements
           and attributes. Note that schema processing is set to lax, i.e. an underlying schema
           (XSD) should be accessible, i.e., all layers should have an corresponding XSD
          associated, the only exception should be the all layer. The priority attribute can be
           used to prioritize annotation levels (in case of overlapping markup during an inline
           markup unification) -- the highest number correspond to the innermost markup. The
          creator attribute can be used to refer to resources that have been provided in an XSF
           file or to a user/agent who has made an annotation level. Keep in mind, that its
          type is NMTOKENS since resources belong to the corpus as a whole, not to a single
          corpusData file</xs:documentation>
281    </xs:annotation>
282    <xs:complexType>
283     <xs:sequence>
284      <xs:choice>
285       <xs:element ref="xsf:meta" minOccurs="0" maxOccurs="unbounded"/>
286       <xs:element ref="xsf:metaRef" minOccurs="0" maxOccurs="unbounded"/>
287      </xs:choice>
288      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
289      <xs:element ref="xsf:log" minOccurs="0"/>
290     </xs:sequence>
291     <xs:attribute ref="xml:id" use="optional"/>
292     <xs:attribute name="priority" type="xs:int" default="0"/>
293     <xs:attribute ref="xsf:creator" use="optional"/>
294    </xs:complexType>
295   </xs:element>
296   <xs:element name="inline">
297    <xs:annotation>
298     <xs:documentation xml:lang="en">The inline element is used as root element for the
          outcome of an XStandoff2Inline transformation.</xs:documentation>
299    </xs:annotation>
300    <xs:complexType>
301     <xs:sequence>
302      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
303     </xs:sequence>
304    </xs:complexType>
305   </xs:element>
306   <xs:element name="milestone">
307    <xs:annotation>
308     <xs:documentation xml:lang="en">The milestone element is modelled after the TEI milestone
           element and should be used when an XStandoff instance should be serialized into an
          inline annotation format. The xsf:segment attribute can be used to refer to the
          segment that was used in the XStandoff representation, the charPos attribute depicts
```

```
        the character position in the primary data where the overlap happened.</
        xs:documentation>
309   </xs:annotation>
310   <xs:complexType>
311    <xs:attribute ref="xsf:segment" use="optional"/>
312    <xs:attribute name="n" use="optional"/>
313    <xs:attribute name="charPos" use="required" type="xs:positiveInteger"/>
314    <xs:attribute name="unit" use="required" type="xs:NMTOKENS"/>
315    <xs:attribute name="mType">
316     <xs:simpleType>
317      <xs:restriction base="xs:string">
318       <xs:enumeration value="start"/>
319       <xs:enumeration value="end"/>
320      </xs:restriction>
321     </xs:simpleType>
322    </xs:attribute>
323   </xs:complexType>
324  </xs:element>
325  <!-- Attribute group declaration -->
326  <xs:attributeGroup name="cd.attributes">
327   <xs:attribute ref="xml:id" use="required"/>
328   <xs:attribute name="xsfVersion" type="xs:decimal" default="1.1"/>
329  </xs:attributeGroup>
330  <xs:attributeGroup name="seg.attributes">
331   <xs:attribute ref="xml:id" use="required"/>
332   <xs:attribute name="primaryData" type="xs:IDREFS" use="optional"/>
333   <xs:attribute name="type" use="optional">
334    <xs:annotation>
335     <xs:documentation xml:lang="en">The type attribute shall be used to specify the segment.
          Possible values are: 'empty' for empty segments depicting formerly empty inline
        elements, 'char' for (non-whitespace) character data of the primary data, 'ws' for
        whitespace character data, 'pun' for punctuation characters, 'dur' when using a
        duration (i.e. non-character based) segmentation, 'seg' for depicting a segment that
         is composed via other segments.</xs:documentation>
336    </xs:annotation>
337    <xs:simpleType>
338     <xs:restriction base="xs:string">
339      <xs:enumeration value="empty"/>
340      <xs:enumeration value="char"/>
341      <xs:enumeration value="ws"/>
342      <xs:enumeration value="pun"/>
343      <xs:enumeration value="dur"/>
344      <xs:enumeration value="seg"/>
345     </xs:restriction>
346    </xs:simpleType>
347   </xs:attribute>
348   <xs:attribute name="content" type="xs:string" use="optional">
349    <xs:annotation>
350     <xs:documentation xml:lang="en">In the very rare case that you want to store the textual
          content of a segment, that is the primary data that is annotated by the annotation
        elements referring to this very segment, use the content attribute. The content can
        be saved in terms of literal characters or in terms of Unicode codepoints. See the
        unicodeNormalizationForm attribute in the latter case.</xs:documentation>
351    </xs:annotation>
352   </xs:attribute>
353   <xs:attribute name="unicodeNormalizationForm" use="optional">
354    <xs:annotation>
355     <xs:documentation xml:lang="en">Some special characters (such as accented characters)
        can be represented in Unicode character sets by unique code points (precomposed) or
        with combining characters (decomposed). Since XSF requires a unique representation,
        a normalized form of Unicode text shall be used if a Unicode character set is used.
        Four Unicode Standard normalization forms are available: Normalization Form D (NFD),
         Normalization Form KD (NFKD), Normalization Form C (NFC), and Normalization Form KC
```

```
        (NFKC). Roughly speaking, NFD and NFKD decompose characters where possible, while
        NFC and NFKC compose characters where possible. Cf. Chapter 2, 'General Structure',
        Chapter 3, 'Conformance' and the Annex #15, 'Unicode Normalization Forms,' of The
        Unicode Standard for details. This information may be used by software processing
        XStandoff instances.</xs:documentation>
356    </xs:annotation>
357    <xs:simpleType>
358     <xs:restriction base="xs:string">
359      <xs:enumeration value="NFD"/>
360      <xs:enumeration value="NFKD"/>
361      <xs:enumeration value="NFC"/>
362      <xs:enumeration value="NFKC"/>
363     </xs:restriction>
364    </xs:simpleType>
365   </xs:attribute>
366   <xs:attribute name="segments" type="xs:IDREFS" use="optional">
367    <xs:annotation>
368     <xs:documentation xml:lang="en">The segments attribute can be used together with the
        mode attribute to create a new segment via referring to already established segments.
        </xs:documentation>
369    </xs:annotation>
370   </xs:attribute>
371   <xs:attribute name="mode" default="continuous">
372    <xs:annotation>
373     <xs:documentation xml:lang="en">The mode attribute marks segments that are created by
        referring to already established segments (cf. the segments attribute) either as
        continuous or disjoint.</xs:documentation>
374    </xs:annotation>
375    <xs:simpleType>
376     <xs:restriction base="xs:string">
377      <xs:enumeration value="continuous"/>
378      <xs:enumeration value="disjoint"/>
379     </xs:restriction>
380    </xs:simpleType>
381   </xs:attribute>
382  </xs:attributeGroup>
383  <xs:attributeGroup name="base.attributes">
384   <xs:attribute name="start" type="xs:nonNegativeInteger" use="optional"/>
385   <xs:attribute name="end" type="xs:nonNegativeInteger" use="optional"/>
386  </xs:attributeGroup>
387  <xs:attributeGroup name="pd.attributes">
388   <xs:attribute name="role" default="master">
389    <xs:annotation>
390     <xs:documentation xml:lang="en">In case of multiple primary data files one primary data
        should be set as master using the role attribute. The master primary data file
        specifies the timeline/character stream that is used for creating segments.</
        xs:documentation>
391    </xs:annotation>
392    <xs:simpleType>
393     <xs:restriction base="xs:NMTOKEN">
394      <xs:enumeration value="master"/>
395      <xs:enumeration value="slave"/>
396     </xs:restriction>
397    </xs:simpleType>
398   </xs:attribute>
399   <xs:attribute name="unit" use="optional" default="chars">
400    <xs:annotation>
401     <xs:documentation xml:lang="en">The unit used for delimiting the segmentation. XSF doesn'
        t use the built-in data type duration, since it lacks some of the units used here.</
        xs:documentation>
402    </xs:annotation>
403    <xs:simpleType>
404     <xs:restriction base="xs:string">
```

```
405       <xs:enumeration value="chars"/>
406       <xs:enumeration value="bytes"/>
407       <xs:enumeration value="microseconds"/>
408       <xs:enumeration value="milliseconds"/>
409       <xs:enumeration value="seconds"/>
410       <xs:enumeration value="minutes"/>
411       <xs:enumeration value="hours"/>
412       <xs:enumeration value="days"/>
413       <xs:enumeration value="weeks"/>
414       <xs:enumeration value="months"/>
415       <xs:enumeration value="years"/>
416       <xs:enumeration value="palFrames"/>
417       <xs:enumeration value="ntscFrames"/>
418       <xs:enumeration value="other"/>
419      </xs:restriction>
420     </xs:simpleType>
421    </xs:attribute>
422    <xs:attribute name="offset" use="optional" type="xs:integer">
423     <xs:annotation>
424      <xs:documentation xml:lang="en">The offset attribute shall be used for specifying an
           offset (in the same unit that is used for delimiting segments) between different
           primaryData files.</xs:documentation>
425     </xs:annotation>
426    </xs:attribute>
427   </xs:attributeGroup>
428   <xs:attributeGroup name="ref.attributes">
429    <xs:attribute name="uri" type="xs:anyURI" use="required">
430     <xs:annotation>
431      <xs:documentation xml:lang="en">The URI specifying the location of the referenced file.</
           xs:documentation>
432     </xs:annotation>
433    </xs:attribute>
434    <xs:attribute name="uriType" use="optional" default="relative">
435     <xs:annotation>
436      <xs:documentation xml:lang="en">This attribute is used to differentiate between relative
            and absolute URIs.</xs:documentation>
437     </xs:annotation>
438     <xs:simpleType>
439      <xs:restriction base="xs:string">
440       <xs:enumeration value="relative"/>
441       <xs:enumeration value="absolute"/>
442      </xs:restriction>
443     </xs:simpleType>
444    </xs:attribute>
445    <xs:attribute name="encoding" type="xs:string" use="optional" default="utf-8"/>
446    <xs:attribute name="unicodeNormalizationForm" use="optional">
447     <xs:annotation>
448      <xs:documentation xml:lang="en">Some special characters (such as accented characters)
           can be represented in Unicode character sets by unique code points (precomposed) or
           with combining characters (decomposed). Since XSF requires a unique representation,
           a normalized form of Unicode text shall be used if a Unicode character set is used.
           Four Unicode Standard normalization forms are available: Normalization Form D (NFD),
            Normalization Form KD (NFKD), Normalization Form C (NFC), and Normalization Form KC
            (NFKC). Roughly speaking, NFD and NFKD decompose characters where possible, while
           NFC and NFKC compose characters where possible. Cf. Chapter 2, 'General Structure',
           Chapter 3, 'Conformance' and the Annex #15, 'Unicode Normalization Forms,' of The
           Unicode Standard for details. These information may be used by software processing
           XStandoff instances.</xs:documentation>
449     </xs:annotation>
450     <xs:simpleType>
451      <xs:restriction base="xs:string">
452       <xs:enumeration value="NFD"/>
453       <xs:enumeration value="NFKD"/>
```

```
      <xs:enumeration value="NFC"/>
      <xs:enumeration value="NFKC"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="mimeType" use="optional">
    <xs:annotation>
     <xs:documentation xml:lang="en">Enumeration of mime-types that can be used for primary
         data files.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="application/x-bytecode.python"/>
      <xs:enumeration value="application/acad"/>
      <xs:enumeration value="application/applefile"/>
      <xs:enumeration value="application/astound"/>
      <xs:enumeration value="application/arj"/>
      <xs:enumeration value="application/base64"/>
      <xs:enumeration value="application/binhex"/>
      <xs:enumeration value="application/binhex4"/>
      <xs:enumeration value="application/book"/>
      <xs:enumeration value="application/cdf"/>
      <xs:enumeration value="application/clariscad"/>
      <xs:enumeration value="application/commonground"/>
      <xs:enumeration value="application/drafting"/>
      <xs:enumeration value="application/dsptype"/>
      <xs:enumeration value="application/dxf"/>
      <xs:enumeration value="application/envoy"/>
      <xs:enumeration value="application/excel"/>
      <xs:enumeration value="application/fractals"/>
      <xs:enumeration value="application/freeloader"/>
      <xs:enumeration value="application/futuresplash"/>
      <xs:enumeration value="application/gnutar"/>
      <xs:enumeration value="application/groupwise"/>
      <xs:enumeration value="application/hlp"/>
      <xs:enumeration value="application/hta"/>
      <xs:enumeration value="application/i-deas"/>
      <xs:enumeration value="application/iges"/>
      <xs:enumeration value="application/iges"/>
      <xs:enumeration value="application/inf"/>
      <xs:enumeration value="application/java"/>
      <xs:enumeration value="application/java-byte-code"/>
      <xs:enumeration value="application/lha"/>
      <xs:enumeration value="application/lzx"/>
      <xs:enumeration value="application/mac-binary"/>
      <xs:enumeration value="application/mac-binhex"/>
      <xs:enumeration value="application/mac-binhex40"/>
      <xs:enumeration value="application/mac-compactpro"/>
      <xs:enumeration value="application/macbinary"/>
      <xs:enumeration value="application/marc"/>
      <xs:enumeration value="application/mbedlet"/>
      <xs:enumeration value="application/mcad"/>
      <xs:enumeration value="application/mime"/>
      <xs:enumeration value="application/mspowerpoint"/>
      <xs:enumeration value="application/msword"/>
      <xs:enumeration value="application/mswrite"/>
      <xs:enumeration value="application/netmc"/>
      <xs:enumeration value="application/octet-stream"/>
      <xs:enumeration value="application/oda"/>
      <xs:enumeration value="application/pdf"/>
      <xs:enumeration value="application/pkcs-12"/>
      <xs:enumeration value="application/pkcs-crl"/>
      <xs:enumeration value="application/pkcs10"/>
```

```
516     <xs:enumeration value="application/pkcs7-mime"/>
517     <xs:enumeration value="application/pkcs7-signature"/>
518     <xs:enumeration value="application/pkix-cert"/>
519     <xs:enumeration value="application/pkix-crl"/>
520     <xs:enumeration value="application/plain"/>
521     <xs:enumeration value="application/postscript"/>
522     <xs:enumeration value="application/powerpoint"/>
523     <xs:enumeration value="application/pro_eng"/>
524     <xs:enumeration value="application/ringing-tones"/>
525     <xs:enumeration value="application/rtf"/>
526     <xs:enumeration value="application/rtf"/>
527     <xs:enumeration value="application/sdp"/>
528     <xs:enumeration value="application/sea"/>
529     <xs:enumeration value="application/set"/>
530     <xs:enumeration value="application/sla"/>
531     <xs:enumeration value="application/smil"/>
532     <xs:enumeration value="application/solids"/>
533     <xs:enumeration value="application/sounder"/>
534     <xs:enumeration value="application/step"/>
535     <xs:enumeration value="application/streamingmedia"/>
536     <xs:enumeration value="application/toolbook"/>
537     <xs:enumeration value="application/vda"/>
538     <xs:enumeration value="application/vnd.fdf"/>
539     <xs:enumeration value="application/vnd.hp-hpgl"/>
540     <xs:enumeration value="application/vnd.hp-pcl"/>
541     <xs:enumeration value="application/vnd.ms-excel"/>
542     <xs:enumeration value="application/vnd.ms-pki.certstore"/>
543     <xs:enumeration value="application/vnd.ms-pki.pko"/>
544     <xs:enumeration value="application/vnd.ms-pki.seccat"/>
545     <xs:enumeration value="application/vnd.ms-pki.stl"/>
546     <xs:enumeration value="application/vnd.ms-powerpoint"/>
547     <xs:enumeration value="application/vnd.ms-project"/>
548     <xs:enumeration value="application/vnd.nokia.configuration-message"/>
549     <xs:enumeration value="application/vnd.nokia.ringing-tone"/>
550     <xs:enumeration value="application/vnd.rn-realmedia"/>
551     <xs:enumeration value="application/vnd.rn-realplayer"/>
552     <xs:enumeration value="application/vnd.wap.wmlc"/>
553     <xs:enumeration value="application/vnd.wap.wmlscriptc"/>
554     <xs:enumeration value="application/vnd.xara"/>
555     <xs:enumeration value="application/vocaltec-media-desc"/>
556     <xs:enumeration value="application/vocaltec-media-file"/>
557     <xs:enumeration value="application/wordperfect"/>
558     <xs:enumeration value="application/wordperfect6.0"/>
559     <xs:enumeration value="application/wordperfect6.1"/>
560     <xs:enumeration value="application/x-123"/>
561     <xs:enumeration value="application/x-aim"/>
562     <xs:enumeration value="application/x-authorware-bin"/>
563     <xs:enumeration value="application/x-authorware-map"/>
564     <xs:enumeration value="application/x-authorware-seg"/>
565     <xs:enumeration value="application/x-bcpio"/>
566     <xs:enumeration value="application/x-binary"/>
567     <xs:enumeration value="application/x-binhex40"/>
568     <xs:enumeration value="application/x-bsh"/>
569     <xs:enumeration value="application/x-bytecode.elisp (compiled elisp)"/>
570     <xs:enumeration value="application/x-bzip"/>
571     <xs:enumeration value="application/x-bzip2"/>
572     <xs:enumeration value="application/x-cdf"/>
573     <xs:enumeration value="application/x-cdlink"/>
574     <xs:enumeration value="application/x-chat"/>
575     <xs:enumeration value="application/x-chat"/>
576     <xs:enumeration value="application/x-cmu-raster"/>
577     <xs:enumeration value="application/x-cocoa"/>
578     <xs:enumeration value="application/x-compactpro"/>
```

```
    <xs:enumeration value="application/x-compress"/>
    <xs:enumeration value="application/x-compressed"/>
    <xs:enumeration value="application/x-conference"/>
    <xs:enumeration value="application/x-cpio"/>
    <xs:enumeration value="application/x-cpt"/>
    <xs:enumeration value="application/x-csh"/>
    <xs:enumeration value="application/x-deepv"/>
    <xs:enumeration value="application/x-director"/>
    <xs:enumeration value="application/x-dvi"/>
    <xs:enumeration value="application/x-elc"/>
    <xs:enumeration value="application/x-envoy"/>
    <xs:enumeration value="application/x-envoy"/>
    <xs:enumeration value="application/x-esrehber"/>
    <xs:enumeration value="application/x-excel"/>
    <xs:enumeration value="application/x-frame"/>
    <xs:enumeration value="application/x-freelance"/>
    <xs:enumeration value="application/x-gsp"/>
    <xs:enumeration value="application/x-gss"/>
    <xs:enumeration value="application/x-gtar"/>
    <xs:enumeration value="application/x-gzip"/>
    <xs:enumeration value="application/x-hdf"/>
    <xs:enumeration value="application/x-helpfile"/>
    <xs:enumeration value="application/x-httpd-imap"/>
    <xs:enumeration value="application/x-ima"/>
    <xs:enumeration value="application/x-internett-signup"/>
    <xs:enumeration value="application/x-inventor"/>
    <xs:enumeration value="application/x-ip2"/>
    <xs:enumeration value="application/x-java-class"/>
    <xs:enumeration value="application/x-java-commerce"/>
    <xs:enumeration value="application/x-javascript"/>
    <xs:enumeration value="application/x-koan"/>
    <xs:enumeration value="application/x-ksh"/>
    <xs:enumeration value="application/x-latex"/>
    <xs:enumeration value="application/x-lha"/>
    <xs:enumeration value="application/x-lisp"/>
    <xs:enumeration value="application/x-livescreen"/>
    <xs:enumeration value="application/x-lotus"/>
    <xs:enumeration value="application/x-lotusscreencam"/>
    <xs:enumeration value="application/x-lzh"/>
    <xs:enumeration value="application/x-lzx"/>
    <xs:enumeration value="application/x-mac-binhex40"/>
    <xs:enumeration value="application/x-macbinary"/>
    <xs:enumeration value="application/x-magic-cap-package-1.0"/>
    <xs:enumeration value="application/x-mathcad"/>
    <xs:enumeration value="application/x-meme"/>
    <xs:enumeration value="application/x-midi"/>
    <xs:enumeration value="application/x-mif"/>
    <xs:enumeration value="application/x-mix-transfer"/>
    <xs:enumeration value="application/x-mplayer2"/>
    <xs:enumeration value="application/x-msexcel"/>
    <xs:enumeration value="application/x-mspowerpoint"/>
    <xs:enumeration value="application/x-navi-animation"/>
    <xs:enumeration value="application/x-navidoc"/>
    <xs:enumeration value="application/x-navimap"/>
    <xs:enumeration value="application/x-navistyle"/>
    <xs:enumeration value="application/x-netcdf"/>
    <xs:enumeration value="application/x-netcdf"/>
    <xs:enumeration value="application/x-newton-compatible-pkg"/>
    <xs:enumeration value="application/x-nokia-9000-communicator-add-on-software"/>
    <xs:enumeration value="application/x-omc"/>
    <xs:enumeration value="application/x-omcdatamaker"/>
    <xs:enumeration value="application/x-omcregerator"/>
    <xs:enumeration value="application/x-pagemaker"/>
```

```
642     <xs:enumeration value="application/x-pcl"/>
643     <xs:enumeration value="application/x-pixclscript"/>
644     <xs:enumeration value="application/x-pkcs10"/>
645     <xs:enumeration value="application/x-pkcs12"/>
646     <xs:enumeration value="application/x-pkcs7-certificates"/>
647     <xs:enumeration value="application/x-pkcs7-certreqresp"/>
648     <xs:enumeration value="application/x-pkcs7-mime"/>
649     <xs:enumeration value="application/x-pkcs7-signature"/>
650     <xs:enumeration value="application/x-pointplus"/>
651     <xs:enumeration value="application/x-portable-anymap"/>
652     <xs:enumeration value="application/x-project"/>
653     <xs:enumeration value="application/x-qpro"/>
654     <xs:enumeration value="application/x-rtf"/>
655     <xs:enumeration value="application/x-sdp"/>
656     <xs:enumeration value="application/x-sea"/>
657     <xs:enumeration value="application/x-seelogo"/>
658     <xs:enumeration value="application/x-sh"/>
659     <xs:enumeration value="application/x-shar"/>
660     <xs:enumeration value="application/x-shockwave-flash"/>
661     <xs:enumeration value="application/x-sit"/>
662     <xs:enumeration value="application/x-sprite"/>
663     <xs:enumeration value="application/x-stuffit"/>
664     <xs:enumeration value="application/x-sv4cpio"/>
665     <xs:enumeration value="application/x-sv4crc"/>
666     <xs:enumeration value="application/x-tar"/>
667     <xs:enumeration value="application/x-tbook"/>
668     <xs:enumeration value="application/x-tcl"/>
669     <xs:enumeration value="application/x-tex"/>
670     <xs:enumeration value="application/x-texinfo"/>
671     <xs:enumeration value="application/x-troff"/>
672     <xs:enumeration value="application/x-troff-man"/>
673     <xs:enumeration value="application/x-troff-me"/>
674     <xs:enumeration value="application/x-troff-ms"/>
675     <xs:enumeration value="application/x-troff-msvideo"/>
676     <xs:enumeration value="application/x-ustar"/>
677     <xs:enumeration value="application/x-visio"/>
678     <xs:enumeration value="application/x-vnd.audioexplosion.mzz"/>
679     <xs:enumeration value="application/x-vnd.ls-xpix"/>
680     <xs:enumeration value="application/x-vrml"/>
681     <xs:enumeration value="application/x-wais-source"/>
682     <xs:enumeration value="application/x-winhelp"/>
683     <xs:enumeration value="application/x-wintalk"/>
684     <xs:enumeration value="application/x-world"/>
685     <xs:enumeration value="application/x-wpwin"/>
686     <xs:enumeration value="application/x-wri"/>
687     <xs:enumeration value="application/x-x509-ca-cert"/>
688     <xs:enumeration value="application/x-x509-user-cert"/>
689     <xs:enumeration value="application/x-zip-compressed"/>
690     <xs:enumeration value="application/xml"/>
691     <xs:enumeration value="application/zip"/>
692     <xs:enumeration value="audio/aiff"/>
693     <xs:enumeration value="audio/basic"/>
694     <xs:enumeration value="audio/it"/>
695     <xs:enumeration value="audio/make"/>
696     <xs:enumeration value="audio/make.my.funk"/>
697     <xs:enumeration value="audio/mid"/>
698     <xs:enumeration value="audio/midi"/>
699     <xs:enumeration value="audio/mod"/>
700     <xs:enumeration value="audio/mpeg"/>
701     <xs:enumeration value="audio/mpeg3"/>
702     <xs:enumeration value="audio/nspaudio"/>
703     <xs:enumeration value="audio/s3m"/>
704     <xs:enumeration value="audio/tsp-audio"/>
```

```
705     <xs:enumeration value="audio/tsplayer"/>
706     <xs:enumeration value="audio/vnd.qcelp"/>
707     <xs:enumeration value="audio/voc"/>
708     <xs:enumeration value="audio/voxware"/>
709     <xs:enumeration value="audio/wav"/>
710     <xs:enumeration value="audio/x-adpcm"/>
711     <xs:enumeration value="audio/x-aiff"/>
712     <xs:enumeration value="audio/x-au"/>
713     <xs:enumeration value="audio/x-gsm"/>
714     <xs:enumeration value="audio/x-jam"/>
715     <xs:enumeration value="audio/x-liveaudio"/>
716     <xs:enumeration value="audio/x-mid"/>
717     <xs:enumeration value="audio/x-midi"/>
718     <xs:enumeration value="audio/x-mod"/>
719     <xs:enumeration value="audio/x-mpeg"/>
720     <xs:enumeration value="audio/x-mpeg-3"/>
721     <xs:enumeration value="audio/x-mpequrl"/>
722     <xs:enumeration value="audio/x-nspaudio"/>
723     <xs:enumeration value="audio/x-pn-realaudio"/>
724     <xs:enumeration value="audio/x-pn-realaudio-plugin"/>
725     <xs:enumeration value="audio/x-psid"/>
726     <xs:enumeration value="audio/x-realaudio"/>
727     <xs:enumeration value="audio/x-twinvq"/>
728     <xs:enumeration value="audio/x-twinvq-plugin"/>
729     <xs:enumeration value="audio/x-vnd.audioexplosion.mjuicemediafile"/>
730     <xs:enumeration value="audio/x-voc"/>
731     <xs:enumeration value="audio/x-wav"/>
732     <xs:enumeration value="audio/xm"/>
733     <xs:enumeration value="chemical/x-pdb"/>
734     <xs:enumeration value="i-world/i-vrml"/>
735     <xs:enumeration value="image/bmp"/>
736     <xs:enumeration value="image/cmu-raster"/>
737     <xs:enumeration value="image/fif"/>
738     <xs:enumeration value="image/florian"/>
739     <xs:enumeration value="image/g3fax"/>
740     <xs:enumeration value="image/gif"/>
741     <xs:enumeration value="image/ief"/>
742     <xs:enumeration value="image/jpeg"/>
743     <xs:enumeration value="image/jutvision"/>
744     <xs:enumeration value="image/naplps"/>
745     <xs:enumeration value="image/pict"/>
746     <xs:enumeration value="image/pjpeg"/>
747     <xs:enumeration value="image/png"/>
748     <xs:enumeration value="image/tiff"/>
749     <xs:enumeration value="image/vasa"/>
750     <xs:enumeration value="image/vnd.dwg"/>
751     <xs:enumeration value="image/vnd.fpx"/>
752     <xs:enumeration value="image/vnd.net-fpx"/>
753     <xs:enumeration value="image/vnd.rn-realflash"/>
754     <xs:enumeration value="image/vnd.rn-realpix"/>
755     <xs:enumeration value="image/vnd.wap.wbmp"/>
756     <xs:enumeration value="image/vnd.xiff"/>
757     <xs:enumeration value="image/x-cmu-raster"/>
758     <xs:enumeration value="image/x-dwg"/>
759     <xs:enumeration value="image/x-icon"/>
760     <xs:enumeration value="image/x-jg"/>
761     <xs:enumeration value="image/x-jps"/>
762     <xs:enumeration value="image/x-niff"/>
763     <xs:enumeration value="image/x-niff"/>
764     <xs:enumeration value="image/x-pcx"/>
765     <xs:enumeration value="image/x-pict"/>
766     <xs:enumeration value="image/x-portable-anymap"/>
767     <xs:enumeration value="image/x-portable-bitmap"/>
```

```
768     <xs:enumeration value="image/x-portable-greymap"/>
769     <xs:enumeration value="image/x-portable-pixmap"/>
770     <xs:enumeration value="image/x-quicktime"/>
771     <xs:enumeration value="image/x-rgb"/>
772     <xs:enumeration value="image/x-tiff"/>
773     <xs:enumeration value="image/x-windows-bmp"/>
774     <xs:enumeration value="image/x-xbitmap"/>
775     <xs:enumeration value="image/x-xbm"/>
776     <xs:enumeration value="image/x-xpixmap"/>
777     <xs:enumeration value="image/x-xwd"/>
778     <xs:enumeration value="image/x-xwindowdump"/>
779     <xs:enumeration value="image/xbm"/>
780     <xs:enumeration value="image/xpm"/>
781     <xs:enumeration value="message/rfc822"/>
782     <xs:enumeration value="model/iges"/>
783     <xs:enumeration value="model/vnd.dwf"/>
784     <xs:enumeration value="model/vrml"/>
785     <xs:enumeration value="model/x-pov"/>
786     <xs:enumeration value="multipart/x-gzip"/>
787     <xs:enumeration value="multipart/x-ustar"/>
788     <xs:enumeration value="multipart/x-zip"/>
789     <xs:enumeration value="music/crescendo"/>
790     <xs:enumeration value="music/x-karaoke"/>
791     <xs:enumeration value="paleovu/x-pv"/>
792     <xs:enumeration value="text/asp"/>
793     <xs:enumeration value="text/css"/>
794     <xs:enumeration value="text/html"/>
795     <xs:enumeration value="text/mcf"/>
796     <xs:enumeration value="text/pascal"/>
797     <xs:enumeration value="text/plain"/>
798     <xs:enumeration value="text/richtext"/>
799     <xs:enumeration value="text/scriplet"/>
800     <xs:enumeration value="text/sgml"/>
801     <xs:enumeration value="text/tab-separated-values"/>
802     <xs:enumeration value="text/uri-list"/>
803     <xs:enumeration value="text/vnd.abc"/>
804     <xs:enumeration value="text/vnd.fmi.flexstor"/>
805     <xs:enumeration value="text/vnd.rn-realtext"/>
806     <xs:enumeration value="text/vnd.wap.wml"/>
807     <xs:enumeration value="text/vnd.wap.wmlscript"/>
808     <xs:enumeration value="text/webviewhtml"/>
809     <xs:enumeration value="text/x-asm"/>
810     <xs:enumeration value="text/x-audiosoft-intra"/>
811     <xs:enumeration value="text/x-c"/>
812     <xs:enumeration value="text/x-component"/>
813     <xs:enumeration value="text/x-fortran"/>
814     <xs:enumeration value="text/x-h"/>
815     <xs:enumeration value="text/x-java-source"/>
816     <xs:enumeration value="text/x-la-asf"/>
817     <xs:enumeration value="text/x-m"/>
818     <xs:enumeration value="text/x-pascal"/>
819     <xs:enumeration value="text/x-script"/>
820     <xs:enumeration value="text/x-script.csh"/>
821     <xs:enumeration value="text/x-script.elisp"/>
822     <xs:enumeration value="text/x-script.guile"/>
823     <xs:enumeration value="text/x-script.ksh"/>
824     <xs:enumeration value="text/x-script.lisp"/>
825     <xs:enumeration value="text/x-script.perl"/>
826     <xs:enumeration value="text/x-script.perl-module"/>
827     <xs:enumeration value="text/x-script.phyton"/>
828     <xs:enumeration value="text/x-script.rexx"/>
829     <xs:enumeration value="text/x-script.scheme"/>
830     <xs:enumeration value="text/x-script.sh"/>
```

```
    <xs:enumeration value="text/x-script.tcl"/>
    <xs:enumeration value="text/x-script.tcsh"/>
    <xs:enumeration value="text/x-script.zsh"/>
    <xs:enumeration value="text/x-server-parsed-html"/>
    <xs:enumeration value="text/x-setext"/>
    <xs:enumeration value="text/x-sgml"/>
    <xs:enumeration value="text/x-speech"/>
    <xs:enumeration value="text/x-uil"/>
    <xs:enumeration value="text/x-uuencode"/>
    <xs:enumeration value="text/x-vcalendar"/>
    <xs:enumeration value="text/xml"/>
    <xs:enumeration value="video/animaflex"/>
    <xs:enumeration value="video/avi"/>
    <xs:enumeration value="video/avs-video"/>
    <xs:enumeration value="video/dl"/>
    <xs:enumeration value="video/fli"/>
    <xs:enumeration value="video/gl"/>
    <xs:enumeration value="video/mpeg"/>
    <xs:enumeration value="video/msvideo"/>
    <xs:enumeration value="video/quicktime"/>
    <xs:enumeration value="video/vdo"/>
    <xs:enumeration value="video/vivo"/>
    <xs:enumeration value="video/vnd.rn-realvideo"/>
    <xs:enumeration value="video/vnd.vivo"/>
    <xs:enumeration value="video/vosaic"/>
    <xs:enumeration value="video/x-amt-demorun"/>
    <xs:enumeration value="video/x-amt-showrun"/>
    <xs:enumeration value="video/x-atomic3d-feature"/>
    <xs:enumeration value="video/x-dl"/>
    <xs:enumeration value="video/x-dv"/>
    <xs:enumeration value="video/x-dv"/>
    <xs:enumeration value="video/x-fli"/>
    <xs:enumeration value="video/x-gl"/>
    <xs:enumeration value="video/x-isvideo"/>
    <xs:enumeration value="video/x-motion-jpeg"/>
    <xs:enumeration value="video/x-mpeg"/>
    <xs:enumeration value="video/x-mpeq2a"/>
    <xs:enumeration value="video/x-ms-asf"/>
    <xs:enumeration value="video/x-ms-asf-plugin"/>
    <xs:enumeration value="video/x-msvideo"/>
    <xs:enumeration value="video/x-qtc"/>
    <xs:enumeration value="video/x-scm"/>
    <xs:enumeration value="video/x-sgi-movie"/>
    <xs:enumeration value="windows/metafile"/>
    <xs:enumeration value="www/mime"/>
    <xs:enumeration value="x-conference/x-cooltalk"/>
    <xs:enumeration value="x-music/x-midi"/>
    <xs:enumeration value="x-world/x-3dmf"/>
    <xs:enumeration value="x-world/x-svr"/>
    <xs:enumeration value="x-world/x-vrml"/>
    <xs:enumeration value="x-world/x-vrt"/>
    <xs:enumeration value="xgl/drawing"/>
    <xs:enumeration value="xgl/movie"/>
    <xs:enumeration value="other"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
</xs:attributeGroup>
<xs:attributeGroup name="imported">
 <xs:annotation>
  <xs:documentation xml:lang="en">This attribute group should be imported into the schema
       files of the respective layers. It consists of the obligatory segment attribute and
       the optional dcrCategory attribute.</xs:documentation>
```

```
892    </xs:annotation>
893    <xs:attribute ref="xsf:segment"/>
894    <xs:attribute ref="xsf:dcrCategory" use="optional"/>
895   </xs:attributeGroup>
896   <!-- Attribute declaration -->
897   <xs:attribute name="segment" type="xs:IDREFS"/>
898   <xs:attribute name="dcrCategory" type="xs:anyURI">
899    <xs:annotation>
900     <xs:documentation xml:lang="en">Use this attribute for providing a link to a ISO Data
           Category Registry category (cf. ISO 12620 for details).</xs:documentation>
901    </xs:annotation>
902   </xs:attribute>
903   <xs:attribute name="previousVersionLogEntry" type="xs:IDREF"/>
904   <xs:attribute name="creator" type="xs:NMTOKENS">
905    <xs:annotation>
906     <xs:documentation xml:lang="en">The creator attribute may be used for referring to (non-)
            human agent (e.g. a resource) that was involved in creating an item.</
           xs:documentation>
907    </xs:annotation>
908   </xs:attribute>
909   <xs:attribute name="affectedItem" type="xs:IDREF"/>
910   <xs:attribute name="alt" type="xs:IDREFS">
911    <xs:annotation>
912     <xs:documentation xml:lang="en">The alt attribute may be used for referring to an
           alternate version, neither in terms of isVersionOf, isBasedOn, isFormatOf or similar
           relations. The type of relation should be specified in the meta data entry of the
           respective element bearing the alt attribute.</xs:documentation>
913    </xs:annotation>
914   </xs:attribute>
915   <!-- complexType declaration -->
916   <xs:complexType name="RefType">
917    <xs:attributeGroup ref="xsf:ref.attributes"/>
918   </xs:complexType>
919  </xs:schema>
```

XQuery-Skript analyseXSF.xq

Das folgende Listing enthält die vollständige aktuelle Fassung des XQUERY-Skripts analyseXSF.xq (vgl. Abschnitt 14.3). Es ist darüber hinaus auch unter der Adresse http://www.xstandoff.net/files/analyzeXSF.zip frei zum Download verfügbar.

Listing 3: analyseXSF.xq

```
(: ########## DOCUMENTATION ##########
  queryXSF.xq - XQuery script for analyzing XStandoff instances
  Version 19.10.2009, 15:00 (GMT)
  (c) Maik Stührenberg; maik.stuehrenberg@uni-bielefeld.de
  This program is free software: you can redistribute it and/or modify it under the terms
      of the GNU Lesser General Public License as published by the Free Software Foundation
      , either version 3 of the License, or (at your option) any later version.
  This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
       without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
      PURPOSE. See the GNU GNU Lesser General Public License for more details.
  You should have received a copy of the GNU GNU Lesser General Public License along with
      this program (file 'lgpl-3.0.txt'). If not, see <http://www.gnu.org/licenses/>.
:)
xquery version "1.0";
declare boundary-space strip;
import schema namespace xsf = "http://www.xstandoff.net/2009/xstandoff/1.1" at "xsf.xsd";
<results>
 {
 for $doc in collection('../data')
 (: declare the target element -- for testing purposes the leaf elements on the first layer
     of the first level are chosen :)
 let $target := $doc//xsf:corpusData/xsf:annotation/xsf:level[1]/xsf:layer[1]/element()[name
     ()!='xsf:meta']//element()[not(*)]
 (: declare the elements to campare :)
 let $compare := $doc//xsf:corpusData/xsf:annotation/xsf:level/xsf:layer//element()
 order by $doc//xsf:corpusData/@xml:id
 return
 <relations docID="{$doc//xsf:corpusData/@xml:id}">
  {
  for $elem in $target
  return
  <targetElement name="{data($elem/name())}" start="{$elem/id(@xsf:segment)/@start}" end="{$
      elem/id(@xsf:segment)/@end}">
   {
   $elem/@*
   }
   {
   (: check for elements with the same segment span :)
   for $virt_same in $compare[@xsf:segment eq $elem/@xsf:segment]
   where not ($virt_same is $elem)
   return
   <identical name="{data($virt_same/name())}" start="{$virt_same/id(@xsf:segment)/@start}"
       end="{$virt_same/id(@xsf:segment)/@end}">
     {
     $virt_same/@*
     }
   </identical>
```

```
39    }
40    {
41    (: check for virtual overlaps :)
42    (: first case - start point identical :)
43    for $virt_overlap in $compare[id(@xsf:segment)/@start eq $elem/id(@xsf:segment)/@start
          and id(@xsf:segment)/@end ne $elem/id(@xsf:segment)/@end]
44    order by $virt_overlap/id(@xsf:segment)/@start
45    return
46    <startPointIdentical name="{$virt_overlap/name()}" start="{$virt_overlap/id(@xsf:segment)
          /@start}" end="{$virt_overlap/id(@xsf:segment)/@end}">
47      {
48      $virt_overlap/@*
49      }
50    </startPointIdentical>
51    }
52    {
53    (: second case - end point identical :)
54    for $virt_overlap in $compare[id(@xsf:segment)/@start ne $elem/id(@xsf:segment)/@start
          and id(@xsf:segment)/@end eq $elem/id(@xsf:segment)/@end]
55    order by $virt_overlap/id(@xsf:segment)/@start
56    return
57    <endPointIdentical name="{$virt_overlap/name()}" start="{$virt_overlap/id(@xsf:segment)/@
          start}" end="{$virt_overlap/id(@xsf:segment)/@end}">
58      {
59      $virt_overlap/@*
60      }
61    </endPointIdentical>
62    }
63    {
64    (: third case - inclusion :)
65    for $virt_overlap in $compare[id(@xsf:segment)/@start lt $elem/id(@xsf:segment)/@start
          and id(@xsf:segment)/@end gt $elem/id(@xsf:segment)/@end]
66    order by $virt_overlap/id(@xsf:segment)/@start
67    return
68    <inclusion name="{$virt_overlap/name()}" start="{$virt_overlap/id(@xsf:segment)/@start}"
          end="{$virt_overlap/id(@xsf:segment)/@end}">
69      {
70      $virt_overlap/@*
71      }
72    </inclusion>
73    }
74    {
75    (: fourth case - overlapping :)
76    for $virt_overlap in $compare[(id(@xsf:segment)/@start gt $elem/id(@xsf:segment)/@start
          and id(@xsf:segment)/@start lt $elem/id(@xsf:segment)/@end and id(@xsf:segment)/@end
          gt $elem/id(@xsf:segment)/@end) or (id(@xsf:segment)/@start lt $elem/id(@xsf:segment)
          /@start and id(@xsf:segment)/@start lt $elem/id(@xsf:segment)/@end and id(@xsf:
          segment)/@start gt $elem/id(@xsf:segment)/@start)]
77    return
78    <overlap name="{$virt_overlap/name()}" start="{$virt_overlap/id(@xsf:segment)/@start}"
          end="{$virt_overlap/id(@xsf:segment)/@end}">
79      {
80      $virt_overlap/@*
81      }
82    </overlap>
83    }
84   </targetElement>
85   }
86  </relations>
87  }
88 </results>
```

Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Algorithmen

Literatur

Zitierte Spezifikationen

Allert, Heidrun, Elke Brenstein, Annika Daun, Gerhard von der Handt, Lars Kilian, Jan Pawlowski, Christoph Richter, Christian Stracke, Maik Stührenberg und Kristina Unverricht (2004). *PAS 1032-2: Aus- und Weiterbildung unter besonderer Berücksichtigung von e-Learning - Teil 2: Didaktisches Objektmodell - Modellierung und Beschreibung didaktischer Szenarien*. Publicly Available Specification (PAS). Berlin: Deutsches Institut für Normung (DIN).

Altheim, Murray, Frank Boumphrey, Sam Dooley, Shane McCarron, Sebastian Schnitzenbaumer und Ted Wugofski (Apr. 2001). *Modularization of XHTML*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410 *[Letzter Abruf: 19.04.2012]*.

ANSI/NISO Z39.85-2007 (Mai 2007). *The Dublin Core Metadata Element Set, Version 1.1*. American National Standard. ANSI/NISO.

Apparao, Vidur, Steve Byrne, Mike Champion, Scott Isaacs, Ian Jacobs, Arnaud Le Hors, Gavin Thomas Nicol, Jonathan Robie, Robert Sutor, Chris Wilson und Lauren Wood (Okt. 1998). *Document Object Model (DOM) Level 1 Specification*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/ *[Letzter Abruf: 19.04.2012]*.

Austin, Daniel, Subramanian Peruvemba, Shane McCarron, Masayasu Ishikawa und Mark Birbeck (Juli 2010). *XHTML Modularization 1.1 – Second Edition*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/REC-xhtml-modularization-20100729 *[Letzter Abruf: 19.04.2012]*.

Axelsson, Jonny, Mark Birbeck, Micah Dubinko, Beth Epperson, Markus Gylling, Masayasua Ishikawa, Shane McCarron, Ann Navarro und Steven Pemberton (Dez. 2010). *XHTML 2.0*. W3C Working Group Note. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/NOTE-xhtml2-20101216 *[Letzter Abruf: 19.04.2012]*.

Beckett, Dave (Feb. 2004). *RDF/XML Syntax Specification (Revised)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ *[Letzter Abruf: 19.04.2012]*.

Beckett, Dave, Eric Miller und Dan Brickley (Juli 2002). *Expressing Simple Dublin Core in RDF/XML*. DCMI Working Draft. Dublin Core Metadata Initiative.
→http://dublincore.org/documents/2002/07/31/dcmes-xml/ *[Letzter Abruf: 19.04.2012]*.

Berglund, Anders, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael Kay, Jonathan Robie und Jérôme Siméon (Jan. 2007). *XML Path Language (XPath). Version 2.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2007/REC-xpath20-20070123/ *[Letzter Abruf: 19. 04. 2012]*.

Berglund, Anders, Mary F. Fernández, Ashok Malhotra, Jonathan Marsh, Marton Nagy und Norman Walsh (Dez. 2010). *XQuery 1.0 and XPath 2.0 Data Model (XDM) (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/REC-xpath-datamodel-20101214/ *[Letzter Abruf: 19. 04. 2012]*.

Berners-Lee, Tim, Roy T. Fielding und Henrik Frystyk Nielsen (Mai 1996). *Hypertext Transfer Protocol – HTTP/1.0*. Request for Comments 1945. Network Working Group.
→http://www.ietf.org/rfc/rfc1945.txt *[Letzter Abruf: 19. 04. 2012]*.

Biron, Paul V. und Ashok Malhotra (Mai 2001). *XML Schema Part 2: Datatypes*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/ *[Letzter Abruf: 19. 04. 2012]*.

– (Okt. 2004). *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/ *[Letzter Abruf: 19. 04. 2012]*.

Boag, Scott, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie und Jérôme Siméon (Jan. 2007). *XQuery 1.0: An XML Query Language*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2007/REC-xquery-20070123/ *[Letzter Abruf: 19. 04. 2012]*.

– (Dez. 2010). *XQuery 1.0: An XML Query Language (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/REC-xquery-20101214/ *[Letzter Abruf: 19. 04. 2012]*.

Bray, Tim, Dave Hollander und Andrew Layman (Jan. 1999). *Namespaces in XML 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1999/REC-xml-names-19990114/ *[Letzter Abruf: 19. 04. 2012]*.

Bray, Tim, Dave Hollander, Andrew Layman und Richard Tobin (Aug. 2006a). *Namespaces in XML 1.0 (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2006/REC-xml-names-20060816 *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2006b). *Namespaces in XML 1.1 (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2006/REC-xml-names11-20060816 *[Letzter Abruf: 19. 04. 2012]*.

Bray, Tim, Jean Paoli und C. M. Sperberg-McQueen (Feb. 1998). *Extensible Markup Language (XML) 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1998/REC-xml-19980210.html *[Letzter Abruf: 19. 04. 2012]*.

Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler und François Yergeau (Feb. 2004). *Extensible Markup Language (XML) 1.1*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-xml11-20040204/ *[Letzter Abruf: 19. 04. 2012]*.

– (Nov. 2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2008/REC-xml-20081126/ *[Letzter Abruf: 19. 04. 2012]*.

Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau und John Cowan (Aug. 2006). *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2006/REC-xml11-20060816 *[Letzter Abruf: 19. 04. 2012]*.

Brickley, Dan und R. V. Guha (Feb. 2004). *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-rdf-schema-20040210/ *[Letzter Abruf: 19. 04. 2012]*.

Burnard, Lou und Syd Bauman, Hrsg. (Okt. 2007). *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Version 1.0.0. Last updated on October 28st 2007. Oxford u. a.: published for the TEI Consortium by Humanities Computing Unit, University of Oxford.
– Hrsg. (März 2011). *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Version 1.9.1. Last updated on March 5th 2011. Oxford u. a.: Published for the TEI Consortium by Humanities Computing Unit, University of Oxford.

Clark, James (Nov. 1999). *XSL Transformations (XSLT) Version 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1999/REC-xslt-19991116 *[Letzter Abruf: 19. 04. 2012]*.

Clark, James und Steven J. DeRose (Nov. 1999). *XML Path Language (XPath). Version 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1999/REC-xpath-19991116/ *[Letzter Abruf: 19. 04. 2012]*.

Cowan, John und Richard Tobin (Okt. 2001). *XML Information Set*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2001/REC-xml-infoset-20011024/ *[Letzter Abruf: 19. 04. 2012]*.
– (Feb. 2004). *XML Information Set (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-xml-infoset-20040204/ *[Letzter Abruf: 19. 04. 2012]*.

Day, Don (Dez. 2010). *Darwin Information Typing Architecture (DITA) Version 1.2*. OASIS Standard. OASIS – Organization for the Advancement of Structured Information Standards.
→http://docs.oasis-open.org/dita/v1.2/os/spec/DITA1.2-spec.html *[Letzter Abruf: 19. 04. 2012]*.

DCMI Usage Board (Okt. 2010). *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative.
→http://www.dublincore.org/documents/2010/10/11/dcmi-terms/ *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J., Ron Jr. Daniel, Paul Grosso, Eve Maler, Jonathan Marsh und Norman Walsh (Aug. 2002). *XML Pointer Language (XPointer)*. W3C Working Draft. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2002/WD-xptr-20020816/ *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J., Ron Jr. Daniel, Eve Maler und Jonathan Marsh (März 2003). *XPointer xmlns() Scheme*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2003/REC-xptr-xmlns-20030325/ *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J., Eve Maler und Ron Jr. Daniel (Dez. 2002). *XPointer xpointer() Scheme*. W3C Working Draft. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2002/WD-xptr-xpointer-20021219/ *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J., Eve Maler und David Orchard (Juni 2001). *XML Linking Language (XLink) Version 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2001/REC-xlink-20010627/ *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J., Eve Maler, David Orchard und Norman Walsh (Mai 2010). *XML Linking Language (XLink) Version 1.1*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/REC-xlink11-20100506/ *[Letzter Abruf: 19. 04. 2012]*.

Dry, Helen Aristar (Apr. 2006). *OLAC Linguistic Subject Vocabulary*. OLAC Recommendation. Open Language Archives Community.
→http://www.language-archives.org/REC/field-20060406.html *[Letzter Abruf: 19. 04. 2012]*.

Expert Advisory Group on Language Engineering Standards (1996). *EAGLES Guidelines*.
→http://www.ilc.cnr.it/EAGLES96/browse.html *[Letzter Abruf: 19. 04. 2012]*.

Fallside, David C. (Mai 2001). *XML Schema Part 0: Primer*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/ *[Letzter Abruf: 19. 04. 2012]*.

Fallside, David C. und Priscilla Walmsley (Okt. 2004). *XML Schema Part 0: Primer Second Edition*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/ *[Letzter Abruf: 19. 04. 2012]*.

Fernández, Mary F., Ashok Malhotra, Jonathan Marsh, Marton Nagy und Norman Walsh (Jan. 2007). *XQuery 1.0 and XPath 2.0 Data Model (XDM)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/ *[Letzter Abruf: 19. 04. 2012]*.

Fielding, Roy T., James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach und Tim Berners-Lee (Juni 1999). *Hypertext Transfer Protocol – HTTP/1.1*. Request for Comments 2616. Network Working Group.
→http://www.ietf.org/rfc/rfc2616.txt *[Letzter Abruf: 19. 04. 2012]*.

Gao, Shudi (Sandy), C. M. Sperberg-McQueen und Henry S. Thompson (Apr. 2012). *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/ *[Letzter Abruf: 19. 04. 2012]*.

Grant, Jan und Dave Beckett (Feb. 2004). *RDF Test Cases*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/ *[Letzter Abruf: 19. 04. 2012]*.

Grosso, Paul, Eve Maler, Jonathan Marsh und Norman Walsh (März 2003a). *XPointer element() Scheme*. W3C Recommendation. World Wide Web Consortium (W3C).

→http://www.w3.org/TR/2003/REC-xptr-element-20030325/ *[Letzter Abruf: 19. 04. 2012]*.

– (März 2003b). *XPointer Framework*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2003/REC-xptr-framework-20030325/ *[Letzter Abruf: 19. 04. 2012]*.

Hayes, Patrick (Feb. 2004). *RDF Semantics*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-rdf-mt-20040210/ *[Letzter Abruf: 19. 04. 2012]*.

Hickson, Ian (Mai 2011). *HTML5. A vocabulary and associated APIs for HTML and XHTML*. W3C Working Draft. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2011/WD-html5-20110525/ *[Letzter Abruf: 19. 04. 2012]*.

Ide, Nancy M. (Aug. 2006). *Linguistic Annotation Framework*. ISO/TC 37/SC4 N311. Genf.

Ide, Nancy M., Greg Priest-Dorman und Jean Véronis (Juni 1996). *Corpus Encoding Standard (CES)*. Techn. Ber. Expert Advisory Group on Language Engineering Standards (EAGLES).
→http://www.cs.vassar.edu/CES/ *[Letzter Abruf: 19. 04. 2012]*.

IEEE Learning Technology Standards Committee WG12 (2002). *Learning Object Metadata (LOM), Version 1.0*. IEEE Standard IEEE 1484.12.1-2002. Institute of Electricial und Electronics Engineers.

ISLE Metadata Initiative (Okt. 2003). *Metadata Elements for Session Descriptions. Version 3.0.4*. Reference Document. Nijmegen: MPI.
→http://www.mpi.nl/IMDI/documents/Proposals/IMDI_MetaData_3.0.4.pdf *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2009). *Metadata Elements for Catalogue Descriptions. Version 3.0.13*. Reference Document. Nijmegen: MPI.
→http://www.mpi.nl/IMDI/documents/Proposals/IMDI_Catalogue_3.0.0.pdf *[Letzter Abruf: 19. 04. 2012]*.

ISO/IEC JTC 1/SC 24 (2009). *Information technology — Computer graphics, image processing and environmental data representation – Extensible 3D (X3D) encodings – Part 1: Extensible Markup Language (XML) encoding*. International Standard ISO/IEC 19776-1:2009. International Organization for Standardization.

ISO/IEC JTC 1/SC 29 (2002). *Information technology — Multimedia content description interface – Part 2: Description definition language*. International Standard ISO/IEC 15938-2:2002. International Organization for Standardization.

ISO/IEC JTC 1/SC 2/WG 2 (2011). *Information technology — Universal Coded Character Set (UCS)*. International Standard ISO/IEC 10646:2011. Genf: International Organization for Standardization/International Electrotechnical Commission.

ISO/IEC JTC 1/SC 32 (2003). *Information technology — Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes*. International Standard ISO/IEC 11179-3:2003. International Organization for Standardization.

ISO/IEC JTC 1/SC 32 (2004a). *Information technology — Metadata registries (MDR) – Part 1: Framework*. International Standard ISO/IEC 11179-1:2004. International Organization for Standardization.
– (2004b). *Information technology — Metadata registries (MDR) – Part 4: Formulation of data definitions*. International Standard ISO/IEC 11179-4:2004. International Organization for Standardization.
– (2004c). *Information technology — Metadata registries (MDR) – Part 6: Registration*. International Standard ISO/IEC 11179-6:2004. International Organization for Standardization.
– (2005a). *Information technology — Metadata registries (MDR) – Part 2: Classification*. International Standard ISO/IEC 11179-2:2005. International Organization for Standardization.
– (2005b). *Information technology — Metadata registries (MDR) – Part 5: Naming and identification principles*. International Standard ISO/IEC 11179-5:2005. International Organization for Standardization.

ISO/IEC JTC 1/SC 34 (1986). *Information Processing — Text and Office Information Systems – Standard Generalized Markup Language*. International Standard ISO 8879. Genf: International Organization for Standardization.
– (Mai 1997a). *Information processing — Hypermedia/Time-based Structuring Language (HyTime) (ISO/IEC 10744:1992 2nd Edition)*. International Standard ISO/IEC 10744:1997. Genf: International Organization for Standardization.
→`http://www1.y12.doe.gov/capabilities/sgml/wg8/document/n1920/` *[Letzter Abruf: 19. 04. 2012]*.
– (1997b). *ISO 8879 TC 2*. Technical Corrigendum ISO/IEC JTC 1/WG 4 N1955. International Organization for Standardization.
→`http://www1.y12.doe.gov/capabilities/sgml/wg8/document/1955.htm` *[Letzter Abruf: 19. 04. 2012]*.
– (2000). *Information technology — Document description and processing languages – HyperText Markup Language (HTML)*. International Standard ISO/IEC 15445:2000. Genf: International Organization for Standardization.
– (2002). *Information technology — Document description and processing languages – Regular Language Description for XML – Part 1: RELAX Core*. International Standard ISO/IEC TR 22250-1:2002. Genf: International Organization for Standardization.
– (2003). *Information technology — Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG (ISO/IEC 19757-2)*. International Standard ISO/IEC 19757-2:2003. Genf: International Organization for Standardization.
– (Jan. 2006a). *Information technology — Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG – Amendment 1: Compact Syntax*. International Standard ISO/IEC 19757-2:2003/Amd 1:2006. Genf: International Organization for Standardization.
– (2006b). *Information technology — Document Schema Definition Language (DSDL) – Part 3: Rule-based validation – Schematron*. International Standard ISO/IEC 19757-3:2006. Genf: International Organization for Standardization.

– (Juni 2006c). *Information technology — Document Schema Definition Language (DSDL) – Part 4: Namespace-based Validation Dispatching Language (NVDL)*. International Standard ISO/IEC 19757-4:2006. Genf: International Organization for Standardization.
– (2008a). *Information technology — Document description and processing languages – Office Open XML File Formats – Part 1: Fundamentals and Markup Language Reference*. International Standard ISO/IEC 29500-1:2008. Genf: International Organization for Standardization.
– (2008b). *Information technology — Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG (ISO/IEC 19757-2)*. International Standard ISO/IEC 19757-2:2008. Genf: International Organization for Standardization.

ISO/IEC JTC 1/SC 36 (2005). *Information technology — Learning, education and training – Quality management, assurance and metrics – Part 1: General approach*. International Standard ISO/IEC 19796-1:2005. International Organization for Standardization.

ISO/IEC JTC 1/SC 7 (Apr. 2005). *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*. International Standard ISO/IEC 19501:2005. Genf: International Organization for Standardization.
– (Nov. 2009a). *Information technology — OMG Unified Modeling Language (OMG UML) Version 2.1.2 – Part 1: Infrastructure*. Draft International Standard ISO/IEC DIS 19505-1. Genf: International Organization for Standardization.
– (Nov. 2009b). *Information technology — OMG Unified Modeling Language (OMG UML) Version 2.1.2 – Part 2: Superstructure*. Draft International Standard ISO/IEC DIS 19505-2. Genf: International Organization for Standardization.

ISO/IEC TC 176/SC 2 (2008). *Quality management systems — Requirements*. International Standard ISO 9001:2008. International Organization for Standardization.

ISO/TC 37/SC 2 (2007a). *Codes for the representation of names of languages – Part 3: Alpha-3 code for comprehensive coverage of languages*. International Standard ISO 639-3. Registration authority: http://www.sil.org/iso639-3/. Genf: International Organization for Standardization.
– (Feb. 2007b). *Presentation/representation of entries in dictionaries – Requirements, recommendations and information*. International Standard ISO 1951:2007. Genf: International Organization for Standardization.

ISO/TC 37/SC 3 (1999). *Computer applications in terminology — Data categories*. Withdrawn International Standard ISO 12620:1999. Genf: International Organization for Standardization.
– (2003a). *Computer applications in terminology — Terminologcial markup framework*. International Standard ISO 16642. Genf: International Organization for Standardization.
– (Okt. 2003b). *Terminology and other language and content resources — Data categories — Part 1: Specification of data categories and management of a data category registry for language resources*. Committee Draft ISO/CD 12620-1 (N 488). Genf: International Organization for Standardization.
→`http://atoll.inria.fr/RNIL/TC37SC4-docs/CD12620-1.pdf` *[Letzter Abruf: 19. 04. 2012]*.

ISO/TC 37/SC 3 (2003c). *Terminology and other language and content resources — Data categories —Part 2: Data category selection (DCS) for electronic terminological resources (ETR)*. Committee Draft ISO/CD 12620-2. Genf: International Organization for Standardization.
→http://atoll.inria.fr/RNIL/TC37SC4-docs/CD12620-2.pdf *[Letzter Abruf: 19. 04. 2012]*.

- (Juli 2004). *Terminology and other language and content resources — Data categories — Part 1: Specification of data categories and management of a data category registry for language resources*. Committee Draft ISO/CD 12620-1 (N 509). Genf: International Organization for Standardization.
→http://lirics.loria.fr/doc_pub/N509_ISO_CD_12620-1_(2004-07-31).doc *[Letzter Abruf: 19. 04. 2012]*.
- (2009). *Terminology and other language and content resources — Specification of data categories and management of a Data Category Registry for language resources*. International Standard ISO 12620:2009. Genf: International Organization for Standardization.

ISO/TC 37/SC 4/WG 1 (Okt. 2005). *Language Resource Management — Feature Structures – Part 1: Feature Structure Representation*. Draft International Standard ISO/DIS 24610-1. Genf: International Organization for Standardization.
→http://www.tc37sc4.org/new_doc/ISO_TC_37-4_N188_Rev5_24610-1_FSR_20051020.pdf *[Letzter Abruf: 19. 04. 2012]*.

- (2006). *Language Resource Management — Feature Structures – Part 1: Feature Structure Representation*. International Standard ISO 24610-1:2006. Genf: International Organization for Standardization.
- (Juli 2008a). *Language Resource Management — Feature Structures – Part 2: Feature System Declaration*. Draft International Standard ISO/DIS 24610-2. Genf: International Organization for Standardization.
- (Mai 2008b). *Language Resource Management — Linguistic Annotation Framework*. Committee Draft ISO/CD 24612. Genf: International Organization for Standardization.
- (März 2009). *Language Resource Management — Linguistic annotation framework (LAF)*. Draft International Standard ISO/DIS 24612. Genf: International Organization for Standardization.
- (2010). *Language Resource Management — Feature Structures – Part 1: Feature Structure Representation*. Committee Draft ISO/CD 24610-1. Genf: International Organization for Standardization.
- (Sep. 2011a). *Language Resource Management — Feature Structures – Part 2: Feature System Declaration*. International Standard ISO 24610-2:2011. Genf: International Organization for Standardization.
- (März 2011b). *Language Resource Management — Feature Structures – Part 2: Feature System Declaration*. Final Draft International Standard ISO/FDIS 24610-2. Genf: International Organization for Standardization.
- (Aug. 2011c). *Language Resource Management — Linguistic annotation framework (LAF)*. Final Draft International Standard ISO/FDIS 24612. Genf: International Organization for Standardization.

ISO/TC 37/SC 4/WG 2 (2005). *Language Resource Management — Morpho-syntactic*

Annotation Framework (MAF). Committee Draft ISO/CD 24611. Genf: International Organization for Standardization.
→http://www.tc37sc4.org/new_doc/ISO_TC_37-4_N225_CD_MAF.pdf *[Letzter Abruf: 19. 04. 2012]*.

- (2008). *Language Resource Management — Morpho-syntactic annotation framework*. Draft International Standard ISO/DIS 24611. Genf: International Organization for Standardization.
- (Juli 2009). *Language Resource Management — Semantic annotation framework (SemAF) – Part 1: Time and events (SemAF-Time, ISO-TimeML)*. Draft International Standard ISO/DIS 24617-1. Genf: International Organization for Standardization.
- (Sep. 2010a). *Language resource management — Semantic annotation framework (SemAF) – Part 4: Semantic roles (SemAF-SRL)*. Working Draft ISO/AWI 24617-4. Genf: International Organization for Standardization.
- (Okt. 2010b). *Language resource management — Semantic annotation framework (SemAF) – Part 5: Discourse structure (SemAF-DS)*. Working Draft ISO/AWI 24617-5. Genf: International Organization for Standardization.
- (2010c). *Language Resource Management — Syntactic annotation framework (SynAF)*. International Standard ISO 24615:2010. Genf: International Organization for Standardization.
- (2010d). *Language Resource Management — Syntactic annotation framework (SynAF)*. Final Draft International Standard ISO/FDIS 24615. Genf: International Organization for Standardization.
  →http://www.tc37sc4.org/new_doc/iso_tc37_sc4_N712_wg2_FDIS_24615_SynAF_June2010-update.pdf *[Letzter Abruf: 19. 04. 2012]*.
- (Jan. 2011). *Language Resource Management — Semantic annotation framework (SemAF) – Part 2: Dialogue acts*. Draft International Standard ISO/DIS 24617-2. Genf: International Organization for Standardization.

ISO/TC 37/SC 4/WG 4 (2008). *Language Resource Management — Lexical markup framework (LMF)*. International Standard ISO 24613:2008. International Organization for Standardization.

ISO TC 46/SC 4 (2009). *The Dublin Core Metadata Element Set, Version 1.1*. International Standard ISO 15836:2009. Genf: International Organization for Standardization.

Johnson, Heidi (Apr. 2006). *OLAC Role Vocabulary*. OLAC Recommendation. Open Language Archives Community.
→http://www.language-archives.org/REC/role-20060406.html *[Letzter Abruf: 19. 04. 2012]*.

Johnson, Heidi und Helen Aristar Dry (Apr. 2006). *OLAC Discourse Type Vocabulary*. OLAC Recommendation. Open Language Archives Community.
→http://www.language-archives.org/REC/discourse-20060406.html *[Letzter Abruf: 19. 04. 2012]*.

Johnston, Pete und Andy Powell (Aug. 2008). *Expressing Dublin Core in HTML/XHTML meta and link elements*. DCMI Recommendation. Dublin Core Metadata Initiative.
→http://dublincore.org/documents/2008/08/04/dc-html/ *[Letzter Abruf: 19. 04. 2012]*.

Kay, Michael (Jan. 2007). *XSL Transformations (XSLT) Version 2.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2007/REC-xslt20-20070123/ *[Letzter Abruf: 19. 04. 2012]*.

– (Mai 2010b). *XSL Transformations (XSLT) Version 2.1*. W3C Working Draft. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/WD-xslt-21-20100511/ *[Letzter Abruf: 19. 04. 2012]*.

Klyne, Graham und Jeremy J. Carroll (Feb. 2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ *[Letzter Abruf: 19. 04. 2012]*.

Kunze, John A. und Thomas Baker (Aug. 2007). *The Dublin Core Metadata Element Set*. Request for Comments 5013. Network Working Group.

Lieske, Christian und Felix Sasaki (Apr. 2007). *Internationalization Tag Set (ITS)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2007/REC-its-20070403/ *[Letzter Abruf: 19. 04. 2012]*.

Malhotra, Ashok, Jim Melton, Norman Walsh und Michael Kay (Dez. 2010). *XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2010/REC-xpath-functions-20101214/ *[Letzter Abruf: 19. 04. 2012]*.

Manola, Frank und Eric Miller (Feb. 2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ *[Letzter Abruf: 19. 04. 2012]*.

Marrin, Chris (Feb. 2011). *WebGL Specification*. Khronos Specification. Khronos Group.
→https://www.khronos.org/registry/webgl/specs/1.0/ *[Letzter Abruf: 19. 04. 2012]*.

Marsh, Jonathan und David Orchard (Dez. 2004). *XML Inclusions (XInclude) Version 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-xinclude-20041220/ *[Letzter Abruf: 19. 04. 2012]*.

Marsh, Jonathan, David Orchard und Daniel Veillard (Nov. 2006). *XML Inclusions (XInclude) Version 1.0 (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2006/REC-xinclude-20061115/ *[Letzter Abruf: 19. 04. 2012]*.

Marsh, Jonathan, Daniel Veillard und Norman Walsh (Sep. 2005). *xml:id Version 1.0*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2005/REC-xml-id-20050909/ *[Letzter Abruf: 19. 04. 2012]*.

NA 009-00-09 AA — Beschreibung und Identifizierung von Dokumenten (Dez. 1995). *Titelangaben von Dokumenten - Teil 3: Verzeichnisse zitierter Dokumente (Literaturverzeichnisse)*. Deutsche Norm DIN 1505-3. Deutsches Institut für Normung (DIN).

Nilsson, Mikael, Andy Powell, Pete Johnston und Ambjörn Naeve (Jan. 2008). *Expressing Dublin Core metadata using the Resource Description Framework*. DCMI Recommendation. Dublin Core Metadata Initiative.
→http://dublincore.org/documents/2008/01/14/dc-rdf/ *[Letzter Abruf: 19. 04. 2012]*.

Pemberton, Steven, Murray Altheim, Daniel Austin, Frank Boumphrey, John Burger, Andrew W. Donoho, Sam Dooley, Klaus Hofrichter, Philipp Hoschka, Masayasu Ishikawa, Warner ten Kate, Peter King, Paula Klante, Shin'ichi Matsui, Shane McCarron, Ann Navarro, Zach Nies, Dave Raggett, Patrick Schmitz, Sebastian Schnitzenbaumer, Peter Stark, Chris Wilson und Dan Zigmond (2000). *XHTML 1.0. The Extensible HyperText Markup Language*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2000/REC-xhtml1-20000126 *[Letzter Abruf: 19. 04. 2012]*.

Pemberton, Steven, Daniel Austin, Jonny Axelsson, Tantek Çelik, Doug Dominiak, Beth Elenbaas, Masayasu Ishikawa, Shin'ichi Matsui, Shane McCarron, Ann Navarro, Subramanian Peruvemba, Rob Relyea, Sebastian Schnitzenbaumer und Peter Stark (2002). *XHTML 1.0. The Extensible HyperText Markup Language (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2002/REC-xhtml1-20020801 *[Letzter Abruf: 19. 04. 2012]*.

Peterson, David, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQueen und Henry S. Thompson (Apr. 2012). *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/ *[Letzter Abruf: 19. 04. 2012]*.

Raggett, Dave (Jan. 1997). *HTML 3.2 Reference Specification*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/REC-html32-19970114 *[Letzter Abruf: 19. 04. 2012]*.

Raggett, Dave, Arnaud Le Hors und Ian Jacobs (Dez. 1999). *HTML 4.01 Specification*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1999/REC-html401-19991224 *[Letzter Abruf: 19. 04. 2012]*.

Reglin, Thomas, Gerhard von der Handt, Susanne Oppitz, Georg Pleger, Stefan Heil, Christian Stracke, Robert Krämer, Thomas Russel, Herbert Müller Philipps Sohn, Claus Bühler, Michael Vennemann, Ulrike Rockmann, Leopold Krause, Jan Pawlowski, Annika Daun, Ulf Ehlers, Christoph Meier, Sybille Hambach, Thomas Berger, Thomas Dudzik, Kai Heddergott, Gerd Neuhaus, Brigitte Strahwald, Kristina Unverricht, Antares Reisky, Heike Wölke und Beate Kramer (2004). *PAS 1032-1: Aus- und Weiterbildung unter besonderer Berücksichtigung von e-Learning - Teil 1: Referenzmodell für Qualitätsmanagement und Qualitätssicherung – Planung, Entwicklung, Durchführung und Evaluation von Bildungsprozessen und Bildungsangeboten*. Publicly Available Specification (PAS). Berlin: Deutsches Institut für Normung (DIN).

Simons, Gary F. und Steven Bird (Apr. 2006). *OLAC Process*. OLAC Standard. Open Language Archives Community.
→http://www.language-archives.org/OLAC/process-20060405.html *[Letzter Abruf: 19. 04. 2012]*.

– (Mai 2008a). *OLAC Metadata*. OLAC Standard. Open Language Archives Community.
→http://www.language-archives.org/OLAC/metadata-20080531.html *[Letzter Abruf: 19. 04. 2012]*.

– (Juli 2008b). *OLAC Repositories*. OLAC Standard. Open Language Archives Community.
→http://www.language-archives.org/OLAC/repositories-20080728.html *[Letzter Abruf: 19. 04. 2012]*.

Sperberg-McQueen, C. M. und Lou Burnard, Hrsg. (Mai 1994). *TEI P3: Guidelines for Electronic Text Encoding and Interchange*. Oxford u. a.: Published for the TEI Consortium by Humanities Computing Unit, University of Oxford.
– Hrsg. (2002). *TEI P4: Guidelines for Electronic Text Encoding and Interchange*. Oxford u. a.: Published for the TEI Consortium by Humanities Computing Unit, University of Oxford.

The Unicode Consortium (1991). *The Unicode Standard. Version 1.0, Volume 1*. Techn. Ber. Reading: The Unicode Consortium.
– (Feb. 2011). *The Unicode Standard. Version 6.0.0*. Techn. Ber. Mountain View: The Unicode Consortium.
→http://www.unicode.org/versions/Unicode6.0.0/ *[Letzter Abruf: 19. 04. 2012]*.

Thompson, Henry S., David Beech, Murray Maloney und Noah Mendelsohn (Mai 2001). *XML Schema Part 1: Structures*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ *[Letzter Abruf: 19. 04. 2012]*.
– (Okt. 2004). *XML Schema Part 1: Structures Second Edition*. W3C Recommendation. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ *[Letzter Abruf: 19. 04. 2012]*.

Weitere zitierte Quellen

Abascal, Rocio, Michel Beigbeder, Aurélien Bénel, Sylvie Calabretto, Bertrand Chabbat, Pierre-Antoine Champin, Noureddine Chatti, David Jouve, Yannick Prié, Béatrice Rumpler und Eric Thivant (Sep. 2003a). „Modéliser la structuration multiple des documents“. In: *Proceedings of the 7th International conference Hypertext, Hypermedia; Products, Tools and Methods (H2PTM'03)*. Paris, S. 253–258.
– (Sep. 2003b). *Modéliser la structuration multiple des documents*. Annexes au rapport final d'activité 2000-2003. Paris: Institut des Sciences du Document Numérique (ISDN).

Abiteboul, Serge, Peter Buneman und Dan Suciu (2000). *Data on the Web: From Relations to Semistructured Data and XML*. San Francisco: Morgan Kaufmann Publishers.

Allsopp, John (März 2007). *Microformats. Empowering Your Markup for Web 2.0*. Apress.

Alt, Susanne (2006). „Data structures for etymology: towards an etymological lexical network“. In: *BULAG* 31, S. 1–12.
→http://hal.archives-ouvertes.fr/hal-00110971/PDF/etymology_final_bulag.pdf.

Ammelburger, Dirk (2002a). „Document Object Model“. In: *XML mit Java ...in 21 Tagen*. Markt+Technik Verlag. Kap. 11, S. 293–330.
– (2002b). „Simple API for XML“. In: *XML mit Java ...in 21 Tagen*. Markt+Technik Verlag. Kap. 9, S. 233–271.

Anderson, Richard, David Baliles, Mark Birbeck, Michael Kay, Steven Livingstone, Brian Loesgen, Didier Martin, Stephen Mohr, Nikola Ozu, Bruce Peat, Jonathan Pinnock, Peter Stark und Kevin Williams (2000). *Professional XML*. Birmingham: Wrox Press.

Ansari, Muhammad Shahid, Norman Zahid und Kyung-Goo Doh (2009). „A Comparative Analysis of XML Schema Languages“. In: *Database Theory and Application. International Conference, DTA 2009, Held as Part of the Future Generation Information Technology Conference, FGIT 2009, Jeju Island, Korea, December 10-12, 2009. Proceedings*. Hrsg. von Dominik Slezak, Tai-hoon Kim, Yanchun Zhang, Jianhua Ma und Kyo-il Chung. Bd. 64. Berlin und Heidelberg: Springer, S. 41–48.

Apache UIMA Development Community (Dez. 2010). *UIMA References*. Version 2.3.1. Apache Software Foundation.

Architecture Committee for the TIPSTER Text Phase II Program (Juni 1996). *TIPSTER Text Phase II Architecture Concept*. Techn. Ber. National Institute of Standards und Technology.
→`http://www-nlpir.nist.gov/related_projects/tipster/docs/con112.doc` *[Letzter Abruf: 19. 04. 2012]*.

Arenas, Marcelo, Wenfei Fan und Leonid Libkin (2002). „What's Hard about XML Schema Constraints“. In: *Proceedings of the 13th International Conference on Database and Expert Systems Applications*. Bd. 2453. Lecture Notes In Computer Science. Berlin und Heidelberg: Springer, S. 269–278.

Balari Ravera, Sergio (1991). „Information-Based Linguistics and Head-Driven Phrase Structure“. In: *Natural Language Processing*. Hrsg. von M. Filgueiras, L. Damas, N. Moreira und A. Tomás. Bd. 476. Lecture Notes in Computer Science. Berlin und Heidelberg: Springer, S. 55–101. DOI: 10.1007/3-540-53678-7_4.

Balmin, Andrey, Yannis Papakonstantinou und Victor Vianu (2004). „Incremental validation of XML documents“. In: *ACM Transactions on Database Systems (TODS)* 29.4, S. 710–751.

Bang-Jensen, Jørgen und Gregory Gutin (2000a). „Acyclic Orderings of the Vertices in Acyclic Digraphs“. In: *Digraphs: Theory, Algorithms and Applications*. 1. Aufl. Springer Monographs in Mathematics. London: Springer. Kap. 4.2, S. 175–176.

– (2000b). „Depth-First Search“. In: *Digraphs: Theory, Algorithms and Applications*. 1. Aufl. Springer Monographs in Mathematics. London: Springer. Kap. 4.1, S. 172–175.

Bański, Piotr (Dez. 2001). *The proposed annotation scheme for the IPI PAN corpus*. IPI PAN Reports 936. Institute of Computer Science, Polish Academy of Sciences (ICS PAS).

– (Aug. 2010). „Why TEI stand-off annotation doesn't quite work: and why you might want to use it nevertheless“. In: *Proceedings of Balisage: The Markup Conference*. Bd. 5. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol5.Banski01.

Bański, Piotr und Adam Przepiórkowski (Aug. 2009). „Stand-off TEI Annotation: the Case of the National Corpus of Polish“. In: *Proceedings of the Third Linguistic Annotation Workshop*. Hrsg. von Manfred Stede, Chu-Ren Huang, Nancy M. Ide und Adam Meyers. Singapur: Association for Computational Linguistics, S. 64–67.
→`http://140.116.245.248/ACL-IJCNLP-2009/LAW-III/pdf/LAW-III11.pdf` *[Letzter Abruf: 19. 04. 2012]*.

– (Juli 2010a). „TEI P5 as a Text Encoding Standard for Multilevel Corpus Annotation“.

In: *Digital Humanities 2010 Conference Abstracts*. The Alliance of Digital Humanities Organisations u. a. London, S. 98–100.
→http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/pdf/ab-616.pdf *[Letzter Abruf: 19. 04. 2012]*.

Bański, Piotr und Adam Przepiórkowski (Mai 2010b). „The TEI and the NCP: the model and its application". In: *Language Resources: From Storyboard to Sustainability and LR Lifecycle Management, Workshop held at the seventh conference on International Language Resources and Evaluation (LREC 2010)*. Hrsg. von Victoria Arranz und Laura van Eerten. Valletta, S. 34–38.

Barillot, Emmanuel und Frédéric Achard (2000). „XML: a lingua franca for science?" In: *Trends in Biotechnology* 18.8, S. 331–333. DOI: 10.1016/S0167-7799(00)01465-7.

Barnard, David T., Lou Burnard, Jean-Pierre Gaspart, Lynne A. Price, C. M. Sperberg-McQueen und Giovanni Battista Varile (1995). „Hierarchical encoding of text: Technical problems and SGML solutions". In: *Computers and the Humanities* 29 (3), S. 211–231. DOI: 10.1007/BF01830617.

Barnard, David T., Ron Hayter, Maria Karababa, George Logan und John McFadden (1988). „SGML-based markup for literary texts: Two problems and some solutions". In: *Computers and the Humanities* 22 (4), S. 265–276. DOI: 10.1007/BF00118602.

Baroni, Marco, Silvia Bernardini, Federica Comastri, Lorenzo Piccioni, Alessandra Volpi, Ra Volpi, Guy Aston und Marco Mazzoleni (Mai 2004). „Introducing the La Repubblica Corpus: A Large, Annotated, TEI(XML)-Compliant Corpus of Newspaper Italian". In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 1771–1774.

Bauman, Syd (Aug. 2005). „TEI HORSEing Around". In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2005/Bauman01/EML2005Bauman01.html *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2008). „Freedom to Constrain: where does attribute constraint come from, mommy?" In: *Proceedings of Balisage: The Markup Conference*. Bd. 1. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol1.Bauman01.

Bauman, Syd, Alejandro Bia, Lou Burnard, Tomaž Erjavec, Christine Ruotolo und Susan Schreibman (Mai 2004). „Migrating Language Resources from SGML to XML: the Text Encoding Initiative Recommendations". In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon.

Bauman, Syd und Terry Catapano (1999). „TEI and the Encoding of the Physical Structure of Books". In: *Computers and the Humanities* 33 (1), S. 113–127. DOI: 10.1023/A:1001769103586.

Bauman, Syd und Julia Flanders (Aug. 2004). „Odd Customizations". In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2004/Bauman01/EML2004Bauman01.html.

Bayerl, Petra Saskia, Harald Lüngen, Daniela Goecke, Andreas Witt und Daniel Naber (2003). „Methods for the semantic analysis of document markup“. In: *Proceedings of the 2003 ACM symposium on Document engineering (DocEng)*. Hrsg. von Cecile Roisin, Ethan Muson und Christine Vanoirbeek. Grenoble: ACM Press, S. 161–170. DOI: 10.1145/958220.958250.

Beer, Florian (Apr. 2008). *Microformats – Semantik für jedermann: Das semantische Web und wie Microformats die Entwicklung vorantreiben werden*. Vdm Verlag.

Behme, Henning und Stefan Mintert (1998). *XML in der Praxis*. 1. Aufl. Addison-Wesley.

– (2000). *XML in der Praxis*. 2. Aufl. Addison-Wesley.
→`http://www.linkwerk.com/pub/xmlidp/2000/` *[Letzter Abruf: 19. 04. 2012]*.

Benedikt, Michael und Christoph Koch (Jan. 2009). „XPath leashed“. In: *ACM Computing Surveys* 41 (1), 3:1–3:54. DOI: 10.1145/1456650.1456653.

Berners-Lee, Tim (Jan. 1997). *Web architecture: Metadata*.
→`http://www.w3.org/DesignIssues/Metadata.html` *[Letzter Abruf: 19. 04. 2012]*.

Berners-Lee, Tim und Mark Fischetti (1999). *Weaving the Web*. New York: HarperCollins.

Berstel, Jean und Luc Boasson (2002). „Formal properties of XML grammars and languages“. In: *Acta Informatica* 38 (9), S. 649–671. DOI: 10.1007/s00236-002-0085-4.

Bird, Steven, Yi Chen, Susan B. Davidson, Haejoong Lee und Yifeng Zheng (2006). „Designing and Evaluating an XPath Dialect for Linguistic Queries“. In: *Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, USA*.

Bird, Steven und Mark Liberman (1999). „Annotation graphs as a framework for multidimensional linguistic data analysis“. In: *Proceedings of the Workshop “Towards Standards and Tools for Discourse Tagging”*. Association for Computational Linguistics, S. 1–10.

– (2001). „A Formal Framework for Linguistic Annotation“. In: *Speech Communication* 33.1–2, S. 23–60.

Bird, Steven und Gary F. Simons (Juli 2001). „The OLAC Metadata Set and Controlled Vocabularies“. In: *Proceedings of the ACL/EACL Workshop on Sharing Tools and Resources for Research and Education*. Association for Computational Linguistics. Toulouse.

– (2003a). „Extending Dublin Core Metadata to Support the Description and Discovery of Language Resources“. In: *Computing and the Humanities* 37.4, S. 12.

– (2003b). „Seven Dimensions of Portability for Language Documentation and Description“. In: *Language* 79, S. 557–582.

Birnbaum, David J. (1997). „In Defense of Invalid SGML“. In: *Proceedings of ACH-ALLC ’97. Joint International Conference of the Association for Computers and the Humanities (ACH) and the Association for Literary & Linguistic Computing (ALLC)*. Queen’s University. Kingston.
→`http://xml.coverpages.org/birnbaumACH97.html` *[Letzter Abruf: 19. 04. 2012]*.

Book, Ronald, Shimon Even, Sheila Greibach und Gene Ott (1971). „Ambiguity in Graphs and Expressions“. In: *IEEE Transactions on Computers* 20, S. 149–153.

Bosak, Jon (Sep. 2001). *The Birth of XML*.
→http://java.sun.com/xml/birth_of_xml.html *[Letzter Abruf: 19. 04. 2012]*.

Bos, Bert (Juli 1995). *SGML-Lite – an easy to parse subset of SGML*.
→http://www.w3.org/People/Bos/Stylesheets/SGML-Lite.html *[Letzter Abruf: 19. 04. 2012]*.

Bouchou, Béatrice, Denio Duarte, Mírian Halfeld Ferrari Alves und Dominique Laurent (2003). „Extending tree automata to model XML validation under element and attribute constraints". In: *Proceedings of ICEIS*. Bd. 1, S. 184–190.

Bourret, Ronald, John Cowan, Ingo Macherius und Simon St. Laurent (Jan. 1999). *Document Definition Markup Language (DDML) Specification, Version 1.0*. W3C Note. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1999/NOTE-ddml-19990119 *[Letzter Abruf: 19. 04. 2012]*.

Box, Don, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte und Dave Winer (Mai 2000). *Simple Object Access Protocol (SOAP) 1.1*. W3C Note. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ *[Letzter Abruf: 19. 04. 2012]*.

Brants, Thorsten (1997). *The NeGra Export Format for Annotated Corpora (Version 3)*. Universität des Saarlandes.
→http://www.coli.uni-saarland.de/~thorsten/publications/Brants-CLAUS98.pdf *[Letzter Abruf: 19. 04. 2012]*.

Bray, Tim (Aug. 1996a). *Draft DD-1996-0001 – Design Principles for XML*.
→http://www.textuality.com/sgml-erb/dd-1996-0001.html *[Letzter Abruf: 19. 04. 2012]*.

– (1996b). *MGML – an SGML Application for Describing Document Markup Languages*. Unpublished draft paper for SGML '96.
→http://www.textuality.com/mgml/index.html *[Letzter Abruf: 19. 04. 2012]*.

– (Nov. 2005). „On Language Creation". In: *Proceedings of the XML 2005 Conference*. Atlanta.

Bray, Tim, Charles Frankston und Ashok Malhotra (Juli 1998). *Document Content Description for XML*. Submission to the World Wide Web Consortium. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/1998/NOTE-dcd-19980731 *[Letzter Abruf: 19. 04. 2012]*.

Broeder, Daan, Thierry Declerck, Laurent Romary, Markus Uneson, Sven Strömqvist und Peter Wittenburg (Mai 2004). „A Large Metadata Domain of Language Resources". In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 369–372.

Broeder, Daan, Marc Kemps-Snijders, Dieter Van Uytvanck, Menzo Windhouwer, Peter Withers, Peter Wittenburg und Claus Zinn (Mai 2010). „A Data Category Registry- and Component-based Metadata Framework". In: *Language Resources: From Storyboard to Sustainability and LR Lifecycle Management, Workshop held at the seventh conference on International Language Resources and Evaluation (LREC 2010)*. Hrsg. von Victoria Arranz und Laura van Eerten. Valletta, S. 43–47.

Broeder, Daan, Oliver Schonefeld, Thorsten Trippel, Dieter van Uytvanck und Andreas Witt (Aug. 2011). „A pragmatic approach to XML interoperability — the Component Metadata Infrastructure (CMDI)“. In: *Proceedings of Balisage: The Markup Conference*. Bd. 7. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol7.Broeder01.

Broeder, Daan und Peter Wittenburg (2006). „The IMDI metadata framework, its current application and future direction“. In: *International Journal of Metadata, Semantics and Ontologies* 1.2, S. 119–132. DOI: 10.1504/IJMSO.2006.011008.

Brown, Martin C. (2002). *XML Processing with Perl, Python, and PHP*. San Francisco: Sybex.

Brüggemann-Klein, Anne (1993a). „Formal Models in Document Processing“. Habilitation. Albert-Ludwig-Universität zu Freiburg i.Br.
→`ftp://ftp.informatik.uni-freiburg.de/papers/brueggem/habil.ps` *[Letzter Abruf: 19.04.2012]*.

– (1993b). „Unambiguity of extended regular expressions in SGML document grammars“. In: *Algorithms—ESA '93*. Hrsg. von Thomas Lengauer. Bd. 726. Lecture Notes in Computer Science. Springer, S. 73–84. DOI: 10.1007/3-540-57273-2_45.

Brüggemann-Klein, Anne und Derick Wood (1992). „Deterministic Regular Languages“. In: *STACS 92. 9th Annual Symposium on Theoretical Aspects of Computer Science Cachan, France, February 13–15, 1992 Proceedings*. Hrsg. von Alain Finkel und Matthias Jantzen. Bd. 577. Lecture Notes in Computer Science. Berlin und Heidelberg: Springer, S. 173–184.

– (1997a). „One-Unambiguous Regular Languages“. In: *Information and computation* 142, S. 182–206.

– (Feb. 1997b). „The Validation of SGML Content Models“. In: *Mathematical and Computer Modelling* 25.4, S. 73–84.

– (1998). *Regular Tree Languages Over Non-Ranked Alphabets*. Unpublished manuscript.
→`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.5397&rep=rep1&type=pdf` *[Letzter Abruf: 19.04.2012]*.

– (2002). *The parsing of extended context-free grammars*. HKUST Theoretical Computer Science Center Research Report HKUST-TCSC-2002-08. The Hong Kong University of Science und Technology Library.

– (Aug. 2004). „Balanced Context-Free Grammars, Hedge Grammars and Pushdown Caterpillar Automata“. In: *Proceedings of Extreme Markup Languages*. Montréal.

Bruno, Emmanuel und Elisabeth Murisasco (2006). „Describing and querying hierarchical XML structures defined over the same textual data“. In: *DocEng '06: Proceedings of the 2006 ACM symposium on Document engineering*. New York: ACM Press, S. 147–154.

Brzozowski, Janusz A. (Okt. 1964). „Derivatives of Regular Expressions“. In: *Journal of the ACM* 11.4, S. 481–494.

Buck, Lee, Charles F. Goldfarb und Paul Prescod (Jan. 2000). *Datatypes for DTDs (DT4DTD) 1.0*. W3C Note. World Wide Web Consortium (W3C).

→http://www.w3.org/TR/2000/NOTE-dt4dtd-20000113 *[Letzter Abruf: 19. 04. 2012]*.

Burnard, Lou (1995). „SGML on the Web: Too Little Too Soon, or Too Much Too Late?" In: *Computers & Texts* 15, S. 12–15.
→http://users.ox.ac.uk/~lou/Belux/ *[Letzter Abruf: 19. 04. 2012]*.

– (Juni 1997). *The Text Encoding Initiative's Recommendations for the Encoding of Language Corpora: Theory and Practice*. Prepared for a seminar on Etiquetación y extracción de información de grandes corpus textuales within the Curso Industrias de la Lengua (14–18 de Julio de 1997). Sponsored by the Fundacion Duques de Soria.
→http://users.ox.ac.uk/~lou/wip/Soria/ *[Letzter Abruf: 19. 04. 2012]*.

– (2005). „Metadata for Corpus Work". In: *Developing Linguistic Corpora: a Guide to Good Practice*. Hrsg. von Martin Wynne. Oxford: Oxbow Books. Kap. 3, S. 30–46.

– (2007). „New tricks from an old dog: An overview of TEI P5". In: *Digital Historical Corpora – Architecture, Annotation, and Retrieval*. Hrsg. von Lou Burnard, Milena Dobreva, Norbert Fuhr und Anke Lüdeling. Dagstuhl Seminar Proceedings 06491. Dagstuhl: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
→http://drops.dagstuhl.de/opus/volltexte/2007/1042/pdf/06491.BurnardLou.Paper.1042.pdf *[Letzter Abruf: 19. 04. 2012]*.

Burnard, Lou und C. M. Sperberg-McQueen (Juni 1995). *TEI Lite: An Introduction to Text Encoding for Interchange*. Revised 2002.
→http://www.tei-c.org/Guidelines/Customization/Lite/teiu5_en.pdf *[Letzter Abruf: 19. 04. 2012]*.

– (Feb. 2006). *TEI Lite: Encoding for Interchange: an introduction to the TEI – Revised for TEI P5 release*.
→http://www.tei-c.org/release/doc/tei-p5-exemplars/html/teilite.doc.html *[Letzter Abruf: 19. 04. 2012]*.

Carey, Brian M. (Sep. 2009). *Meet CAM: A new XML validation technology. Take semantic and structural validation to the next level*.
→http://www.ibm.com/developerworks/xml/library/x-cam/?S_TACT=105AGX54&S_CMP=C0924&ca=dnw-1036&ca=dth-x&open&cm_mmc=6015-_-n-_-vrm_newsletter-_-10731_131528&cmibm_em=dm:0:13962324 *[Letzter Abruf: 19. 04. 2012]*.

Carletta, Jean, Stefan Evert, Ulrich Heid und Jonathan Kilgour (Dez. 2005). „The NITE XML Toolkit: data model and query language". In: *Language Resources and Evaluation* 39.4, S. 313–334.

Carletta, Jean, Stefan Evert, Jonathan Kilgour, Craig Nicol, Dennis Reidsma, Judy Robertson und Holger Voormann (Juni 2009). *Documentation for the NITE XML Toolkit*. Revision 0.3.
→http://groups.inf.ed.ac.uk/nxt/documentation/ *[Letzter Abruf: 19. 04. 2012]*.

Carletta, Jean und Amy Isard (Juni 1999). „The MATE Annotation Workbench: User Requirements". In: *Proceedings of the ACL Workshop: Towards Standards and Tools for Discourse Tagging*. Hrsg. von Marilyn Walker. Somerset, S. 11–17.

Caron, Pascal und Djelloul Ziadi (2000). „Characterization of Glushkov automata“. In: *Theoretical Computer Science* 233.1–2, S. 75–90. DOI: 10.1016/S0304-3975(97)00296-X.

Carpenter, Bob (1992). *The Logic of Typed Feature Structures*. Cambridge: Cambridge University Press.

Carroll, Lewis (2003). *Alice's Adventures in Wonderland and Through the Looking-Glass*. Reissue. Penguin Classics.

Carstensen, Kai-Uwe (Mai 2003). „Rezension zu Willée, Gerd/Schröder, Bernhard/Schmitz, Hans-Christian (eds.) (2002): Computerlinguistik. Was geht, was kommt? Sankt Augustin. (Sprachwissenschaft, Computerlinguistik und Neue Medien 4)“. In: *Linguistik Online* 17, S. 155–161.

Caton, Paul (Dez. 2000). „Markup's current imbalance“. In: *Markup Languages – Theory & Practice* 3 (1), S. 1–13.

Chatti, Noureddine, Sylvie Calabretto und Jean Marie Pinon (2004). „Vers un environment de gestion de documentes à structures multiples“. In: *20èmes Journées Bases de Données Avancées, BDA '04, Montpellier, 19 - 22 octobre 2004, Actes (Informal Proceedings)*. Hrsg. von Jacques Le Maitre, S. 47–64.

Chatti, Noureddine, Suha Kaouk, Sylvie Calabretto und Jean Marie Pinon (Aug. 2007). „MultiX: an XML based formalism to encode multi-structured documents“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2007/Chatti01/EML2007Chatti01.html` *[Letzter Abruf: 19. 04. 2012]*.

Chen, Haiming und Ping Lu (2011). „Assisting the design of XML Schema: diagnosing nondeterministic content models“. In: *Proceedings of the 13th Asia-Pacific Web Conference (APWeb 2011)*. Bd. 6612. Lecture Notes In Computer Science. Springer.

Chomsky, Noam (1955). „Logical Syntax and Semantics: Their Linguistic Relevance“. In: *Language* 31.1, S. 36–45.

– (1956). „Three Models for the Description of Language“. In: *IRE Transactions on Information Theory* 2, S. 113–124.

– (1959). „On certain formal properties of grammars“. In: *Information and Control* 2.2, S. 137–167. DOI: 10.1016/S0019-9958(59)90362-6.

Cieri, Christopher, Khalid Choukri, Nicoletta Calzolari, D. Terence Langendoen, Johannes Leveling, Martha Palmer, Nancy M. Ide und James Pustejovsky (Mai 2010). „A Road Map for Interoperable Language Resource Metadata“. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner und Daniel Tapias. Valletta: European Language Resources Association (ELRA), S. 2506–2509.

CLARIN (2009a). *Concept Registry Service*.
→`http://www.clarin.eu/files/concept_registry-CLARIN-ShortGuide.pdf` *[Letzter Abruf: 19. 04. 2012]*.

– (2009b). *Standards for Text Encoding*.

→http://www.clarin.eu/files/standards-text-CLARIN-ShortGuide.pdf *[Letzter Abruf: 19. 04. 2012]*.

Clark, James (Dez. 1997). *Comparison of SGML and XML*. W3C Note. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/NOTE-sgml-xml-971215 *[Letzter Abruf: 19. 04. 2012]*.

– (Nov. 2001a). *The Design of RELAX NG*.
→http://www.thaiopensource.com/relaxng/design.html *[Letzter Abruf: 19. 04. 2012]*.

– (Feb. 2001b). *TREX – Tree Regular Expressions for XML Language Specification*.
→http://www.thaiopensource.com/trex/spec.html *[Letzter Abruf: 19. 04. 2012]*.

– (Feb. 2002). *An algorithm for RELAX NG validation*.
→http://www.thaiopensource.com/relaxng/derivative.html *[Letzter Abruf: 19. 04. 2012]*.

– (Nov. 2010). *XML vs the Web*.
→http://blog.jclark.com/2010/11/xml-vs-web_24.html *[Letzter Abruf: 19. 04. 2012]*.

Clark, James, John Cowan und Makoto Murata (März 2003). *RELAX NG Compact Syntax Tutorial*. Working Draft. OASIS – Organization for the Advancement of Structured Information Standards.
→http://relaxng.org/compact-tutorial-20030326.html *[Letzter Abruf: 19. 04. 2012]*.

Clark, James und Kohsuke Kawaguchi (Sep. 2001a). *Guidelines for using W3C XML Schema Datatypes with RELAX NG*. Committee Specification. OASIS – Organization for the Advancement of Structured Information Standards.
→http://www.relaxng.org/xsd-20010907.html *[Letzter Abruf: 19. 04. 2012]*.

– (Dez. 2001b). *RELAX NG DTD Compatibility*. Committee Specification. OASIS – Organization for the Advancement of Structured Information Standards.
→http://www.relaxng.org/compatibility-20011203.html *[Letzter Abruf: 19. 04. 2012]*.

Clark, James und Makoto Murata (Dez. 2001). *RELAX NG Tutorial*. Committee Specification. OASIS – Organization for the Advancement of Structured Information Standards.
→http://relaxng.org/tutorial-20011203.html *[Letzter Abruf: 19. 04. 2012]*.

Clément, Lionel und Èric Villemonte de la Clergerie (2005). „MAF: a Morphosyntactic Annotation Framework“. In: *Proceedings of the 2nd Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Poznan, S. 90–94.

Cole, Ron, Joseph Mariani, Hans Uszkoreit, Giovanni Battista Varile, Annie Zaenen, Antonio Zampolli und Victor Zue, Hrsg. (1997). *Survey of the State of the Art in Human Language Technology*. Studies in Natural Language Processing. Cambridge University Press.

Comon, Hubert, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison und Marc Tommasi (Nov. 2008). *Tree Automata Techniques and Applications*.
→http://www.grappa.univ-lille3.fr/tata *[Letzter Abruf: 19. 04. 2012]*.

Connolly, Dan, Rohit Khare und Adam Rifkin (Sep. 1997). *The Evolution of Web Documents: The Ascent of XML*.

→http://xml.coverpages.org/connollyAscent.html *[Letzter Abruf: 19. 04. 2012]*.

Coombs, James H., Allen H. Renear und Steven J. DeRose (1987). „Markup Systems and the Future of Scholarly Text Processing“. In: *Communications of the ACM* 30.11, S. 933–947.

Copestake, Ann (2002). *Implementing Typed Feature Structure Grammars*. Stanford: CSLI Publications.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest und Clifford Stein (2007). *Algorithmen – eine Einführung*. 2. Aufl. Oldenbourg Wissenschaftsverlag.

Costello, Roger L. (Nov. 2011). *What Kind Of Thing Is It? - A valuable lesson learned on the difference between XML Schemas and ontologies*.
→http://xfront.com/What-Kind-Of-Thing-Is-It.pdf *[Letzter Abruf: 19. 04. 2012]*.

Costello, Roger L. und Robin A. Simmons (Jan. 2008). *Tutorials on Schematron*.
→http://www.xfront.com/schematron/index.html *[Letzter Abruf: 19. 04. 2012]*.

Courcelle, Bruno (1989). „On recognizable sets and tree automata“. In: *Resolution of Equations in Algebraic Structures, Algebraic Techniques*. Hrsg. von Maurice Nivat und Hassan Aït-Kaci. Academic Press.

Cowan, John, Jeni Tennison und Wendell Piez (Aug. 2006). „LMNL Update“. In: *Proceedings of Extreme Markup Languages*. Montréal.

Cummings, James und Sebastian Rahtz (Juni 2011). „Introductory TEI ODD“. In: *Digital Humanities 2011 Conference Abstracts*. The Alliance of Digital Humanities Organisations u. a. Stanford University.
→https://www.stanford.edu/group/dh2011/cgi-bin/wordpress/wp-content/uploads/2011/05/DH2011_BookOfAbs.pdf *[Letzter Abruf: 19. 04. 2012]*.

Cunningham, Hamish (Mai 2002). „GATE, a General Architecture for Text Engineering“. In: *Computers and the Humanities* 36.2, S. 223–254.

Dalby, David, Lee Gillam, Christopher Cox und Debbie Garside (Mai 2004). „Standards for Language Codes: Developing ISO 639“. In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 127–130.

Dalrymple, Mary (2001). *Lexical Functional Grammar*. Bd. 34. Syntax and Semantic. New York: Academic Press.

Dal Zilio, Silvano und Denis Lugiez (2003). „XML Schema, Tree Logic and Sheaves Automata“. In: *Rewriting Techniques and Applications*. Hrsg. von Robert Nieuwenhuis. Bd. 2706. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 246–263. DOI: 10.1007/3-540-44881-0_18.

Davidson, Andrew, Matthew Fuchs, Mette Hedin, Mudita Jain, Jari Koistinen, Chris Lloyd, Murray Maloney und Kelly Schwarzhof (Juli 1999). *Schema for Object-Oriented XML 2.0*. W3C Note. World Wide Web Consortium (W3C).
→http://www.w3.org/TR/NOTE-SOX/ *[Letzter Abruf: 19. 04. 2012]*.

Declerck, Thierry (Mai 2006). „SynAF: Towards a Standard for Syntactic Annotation“. In: *Proceedings of the Fifth International Language Resources and Evaluation (LREC 2006)*. European Language Resources Association (ELRA). Genua, S. 229–232.

– (Mai 2008). „A Framework for Standardized Syntactic Annotation“. In: *Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis und Daniel Tapias. European Language Resources Association (ELRA). Marrakech, S. 3025–3028.
→http://www.lrec-conf.org/proceedings/lrec2008/pdf/770_paper.pdf *[Letzter Abruf: 19. 04. 2012]*.

Declerck, Thierry, Mirjam Kessler und Ulrich Krieger (Jan. 2006). *Evaluation of initiatives for morpho-syntactic and syntactic annotation*. 3.1. Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI).
→http://lirics.loria.fr/doc_pub/Del3_1_V2.pdf *[Letzter Abruf: 19. 04. 2012]*.

Dehmer, Matthias, Alexander Mehler und Frank Emmert-Streib (2007). „Graph-theoretical Characterizations of Generalized Trees“. In: *Proceedings of the 2007 International Conference on Machine Learning: Models, Technologies & Applications (MLMTA'07), June 25-28, 2007, Las Vegas*, S. 113–117.

Dekhtyar, Alex und Ionut Emil Iacob (Feb. 2005). „A framework for management of concurrent XML markup“. In: *Data & Knowledge Engineering* 52.2, S. 185–208.

De la Clergerie, Èric Villemonte (Mai 2004). *Some linguistic examples of Feature Structures*. ISO/TEI Feature Structures Workgroup.
→http://www.tei-c.org/Activities/Workgroups/FS/FS-examples.pdf *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J. (1997). *The SGML FAQ Book. Understanding the Foundation of HTML and XML*. Electronic Publishing Series. Boston, Dordrecht und London: Kluwer Academic Publishers.

– (Dez. 1999). „XML Linking“. In: *ACM Computing Survey* 31.4es.

– (Aug. 2004). „Markup Overlap: A Review and a Horse“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2004/DeRose01/EML2004DeRose01.html *[Letzter Abruf: 19. 04. 2012]*.

DeRose, Steven J., David G. Durand, Elli Mylonas und Allen H. Renear (1990). „What is text, really?“ In: *Journal of Computing in Higher Education* 1.2, S. 3–26.

Diewald, Nils, Maik Stührenberg, Anna Garbar und Daniela Goecke (2008). „Serengeti – Webbasierte Annotation semantischer Relationen“. In: *Journal for Language Technology and Computational Linguistics* 23.2, S. 74–93.

Di Iorio, Angelo, Silvio Peroni und Fabio Vitali (Aug. 2009). „Towards markup support for full GODDAGs and beyond: the EARMARK approach“. In: *Proceedings of Balisage: The Markup Conference*. Bd. 3. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol3.Peroni01.

Dijkstra, Edsger W. (März 1968). „Go To Statement Considered Harmful“. In: *Communications of the ACM* 11.3, S. 147–148.

DIN Deutsches Institut für Normung e. V. (Feb. 2009). *Richtlinie für Normenausschüsse*. →http://www.din.de/sixcms_upload/media/2896/Richtlinie_NA.pdf *[Letzter Abruf: 19. 04. 2012]*.

– (2010). *Übersicht der Normenausschüsse*. →http://www.din.de/cmd?workflowname=DinNaSearch&menuid=47420&cmsareaid=47420&cmsrubid=47445&menurubricid=47445&languageid=de *[Letzter Abruf: 19. 04. 2012]*.

Dipper, Stefanie (2005). „XML-based Stand-off Representation and Exploitation of Multi-Level Linguistic Annotation“. In: *Proceedings of Berliner XML Tage 2005 (BXML 2005)*. Belin, S. 39–50. →http://www.ling.uni-potsdam.de/~dipper/papers/xmltage05.pdf *[Letzter Abruf: 19. 04. 2012]*.

Dipper, Stefanie und Michael Götze (2005). „Accessing Heterogeneous Linguistic Data – Generic XML-based Representation and Flexible Visualization“. In: *Proceedings of the 2nd Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Poznan, S. 206–210.

Dipper, Stefanie, Michael Götze, Uwe Küssner und Manfred Stede (2007). „Representing and Querying Standoff XML“. In: *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Data Structures for Linguistic Resources and Applications. Proceedings of the Biennial GLDV Conference 2007*. Hrsg. von Georg Rehm, Andreas Witt und Lothar Lemnitzer. Tübingen: Gunter Narr Verlag, S. 337–346.

Dipper, Stefanie, Michael Götze, Manfred Stede und Tillmann Wegst (2004). „ANNIS: A Linguistic Database for Exploring Information Structure“. In: *Interdisciplinary Studies on Information Structure*. Hrsg. von Shinichiro Ishihara, Micheala Schmitz und Anne Schwarz. Bd. 1. ISIS Working Papers of the SFB 632. Potsdam: Universitätsverlag Potsdam, S. 245–279.

Dipper, Stefanie, Hannah Kermes, Esther König-Baumer, Wolfgang Lezius, Frank H. Müller und Tylman Ule (2002). *DEREKO (DEutsches REferenzKOrpus) German Reference Corpus*. Final Report (Part I). Seminar für Sprachwissenschaft, Universität Tübingen.

Dörre, Jochen (Juli 1997). „Efficient Construction of Underspecified Semantics under Massive Ambiguity“. In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, S. 386–393. DOI: 10.3115/976909.979667.

Dunlop, Dominic (1995). „Practical considerations in the use of TEI headers in a large corpus“. In: *Computers and the Humanities* 29 (1), S. 85–98. DOI: 10.1007/BF01830318.

Durand, David G. (1999). „Palimpsest: Change-Oriented Concurrency Control for the Support of Collaborative Applications“. Diss. Boston University Graduate School of Arts und Sciences.

Durusau, Patrick und Matthew Brook O'Donnell (Aug. 2002a). „Coming Down From The Trees: Next step in the Evolution of Markup?“ In: *Proceedings of Extreme Markup*

Languages. Wird auch zitiert als: Just-In-Time-Trees (JITTs): Next Step in the Evolution of Markup? Montréal.

Durusau, Patrick und Matthew Brook O'Donnell (2002b). „Concurrent Markup for XML Documents". In: *Proceedings of the XML Europe 2002*. Barcelona.
→`https://ssl.bnt.com/idealliance/papers/xmle02/dx_xmle02/papers/03-03-07/03-03-07.pdf` *[Letzter Abruf: 19. 04. 2012]*.

– (Dez. 2002c). „Implementing JITTs (Just-in-Time-Trees)". In: *Proceedings of XML 2002 conference*. Baltimore.

– (Aug. 2005). „Tabling the Overlap Discussion". In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2004/Durusau01/EML2004Durusau01.html` *[Letzter Abruf: 19. 04. 2012]*.

Dybkjær, Laila und Niels Ole Bernsen (Okt. 2000a). „The MATE Markup Framework". In: *Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue*. Association for Computational Linguistics. Hong Kong, S. 19–28. DOI: 10.3115/1117736.1117739.

– (Okt. 2000b). „Towards Corpus Annotation Standards. The MATE Workbench". In: *Proceedings of the COCOSDA Beijing Meeting*. The International Committee for the Co-ordination, Standardisation of Speech Databases und Assesment Techniques for Speech Input/Output. Beijing.

Eckart, Richard (2008). „Choosing an XML database for linguistically annotated corpora". In: *SDV – Sprache und Datenverarbeitung* 32.1, S. 7–22.

Eckart, Richard und Elke Teich (2007). „An XML-Based Data Model for Flexible Representation and Query of Linguistically Interpreted Corpora". In: *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Data Structures for Linguistic Resources and Applications. Proceedings of the Biennial GLDV Conference 2007*. Hrsg. von Georg Rehm, Andreas Witt und Lothar Lemnitzer. Tübingen: Gunter Narr Verlag, S. 327–337.

Erjavec, Tomaž und Tamás Váradi (2001). „The TELRI tool catalogue: structure and prospects". In: *Proceedings of the ACL/EACL Workshop on Sharing Tools and Resources for Research and Education*. Bd. 15. Stroudsburg: Association for Computational Linguistics, S. 19–22d. DOI: 10.3115/1118062.1118066.

Evert, Stefan, Jean Carletta, Timothy J. O'Donnell, Jonathan Kilgour, Andreas Vögele und Holger Voormann (März 2003). *The NXT Object Model*. 2.1.

Evjen, Bill, Kent Sharkey, Thiru Thangarathinam, Michael Kay, Alessandro Vernet und Sam Ferguson (2007a). *Professional XML*. Indianapolis: Wiley Publishing.

– (2007b). „Simple API for XML (SAX)". In: *Professional XML*. Indianapolis: Wiley Publishing. Kap. 13, S. 377–403.

– (2007c). „XML and Java". In: *Professional XML*. Indianapolis: Wiley Publishing. Kap. 16, S. 471–500.

– (2007d). „XML Document Object Model (DOM)". In: *Professional XML*. Indianapolis: Wiley Publishing. Kap. 12, S. 353–376.

Fankhauser, Peter (Sep. 2001). „XQuery formal semantics state and challenges“. In: *ACM SIGMOD Record* 30 (3), S. 14–19. DOI: 10.1145/603867.603870.

Fankhauser, Peter und Patrick Lehti (2003). „IPSI-XQ: Eine Implementierung von XQuery und seiner formalen Semantik.“ In: *it - Information Technology* 45.3, S. 133–136. DOI: 10.1524/itit.45.3.133.20196.

Fan, Wenfei und Leonid Libkin (Mai 2002). „On XML integrity constraints in the presence of DTDs“. In: *Journal of the ACM* 49.3, S. 368–406. DOI: 10.1145/567112.567117.

Farrar, Scott und D. Terence Langendoen (2003). „A Linguistic Ontology for the Semantic Web“. In: *GLOT International* 7.3, S. 97–100.

Fielding, Thomas Roy (2000). „Architectural Styles and the Design of Network-based Software Architectures“. Dissertation. University of California, Irvine.
→`http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Fillmore, Charles J., Nancy M. Ide, Daniel Jurafsky und Catherine Macleod (1998). „An American National Corpus: A Proposal“. In: *Proceedings of the First International Language Resources and Evaluation (LREC 1998)*. European Language Resources Association (ELRA). Granada.

Fiorello, Davide, Nicola Gessa, Paolo Marinelli und Fabio Vitali (Aug. 2004). „DTD++ 2.0: Adding support for co-constraints“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2004/Vitali01/EML2004Vitali01.html` *[Letzter Abruf: 19. 04. 2012]*.

Flanders, Julia und Syd Bauman (Juli 2010). „Using ODD for Multi-purpose TEI Documentation“. In: *Digital Humanities 2010 Conference Abstracts*. The Alliance of Digital Humanities Organisations u. a. London.
→`http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/pdf/ab-750.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Fong, Joseph, San Kuen Cheung und Herbert Shiu (März 2008). „The XML Tree Model — toward an XML conceptual schema reversed from XML Schema Definition“. In: *Data & Knowledge Engineering* 64.3, S. 624–661.

Francopoulo, Gil, Nuria Bel, Monte George, Nicoletta Calzolari, Monica Monachini, Mandy Pet und Claudia Soria (2006). „Lexical Markup Framework (LMF) for NLP Multilingual Resources“. In: *Proceedings of the Workshop On Multilingual Language Resources And Interoperability*.

– (2009). „Multilingual resources for NLP in the lexical markup framework (LMF)“. In: *Language Resources and Evaluation* 43 (1), S. 57–70. DOI: 10.1007/s10579-008-9077-5.

Francopoulo, Gil, Thierry Declerck, Monica Monachini und Laurent Romary (Mai 2006). „The relevance of standards for research infrastructures“. In: *Proceedings of the Fifth International Language Resources and Evaluation (LREC 2006)*. European Language Resources Association (ELRA). Genua.

Francopoulo, Gil, Thierry Declerck, Virach Sornlertlamvanich, Èric Villemonte de lade la Clergerie und Monica Monachini (Mai 2008). „Data Category Registry: morpho-syntactic and syntactic profiles“. In: *Proceedings of the LREC 2008 Workshop "Uses and usage of language resource-related standards"*. Hrsg. von Andreas Witt, Felix Sasaki, Elke Teich, Nicoletta Calzolari und Peter Wittenburg. ELRA/ELDA.

Frankston, Charles und Henry S. Thompson (Juli 1998). *XML-Data reduced*. Draft. Microsoft & University of Edinburgh.
→`http://www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm` *[Letzter Abruf: 19. 04. 2012]*.

Freisler, Stefan (1994). „Hypertext – Eine Begriffsbestimmung“. In: *Deutsche Sprache* 22.1, S. 19–50.

Fuchs, Matthew und Allen Brown (Aug. 2003). „Supporting UPA and restriction on an extension of XML Schema“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2003/Fuchs01/EML2003Fuchs01.html` *[Letzter Abruf: 19. 04. 2012]*.

Fuchs, Matthew, Murray Maloney und Alex Milowski (Sep. 1998). *Schema for Object-Oriented XML*. Submission to the World Wide Web Consortium. World Wide Web Consortium (W3C).
→`http://www.w3.org/TR/1998/NOTE-SOX-19980930/` *[Letzter Abruf: 19. 04. 2012]*.

Gansner, Emden, Eleftherios Koutsofios und Stephen North (Jan. 2006). *Drawing graphs with dot*.
→`http://www.graphviz.org/Documentation/dotguide.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Garside, Roger, Geoffrey Leech und Anthony McEnery, Hrsg. (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.

Gazdar, Gerald, Ewan Klein, G. K. Pullum und Ivan A. Sag (1985). *Generalized Phrase Structure Grammar*. Cambridge: Harvard University Press.

Gécseg, Ferenc und Magnus Steinby (1997). „Tree Languages“. In: *Handbook of Formal Languages*. Hrsg. von Grzegorz Rozenberg und Arto Salomaa. Bd. 3: Beyond Words. Berlin u. a.: Springer-Verlag, S. 1–68.

Gelade, Wouter, Tomasz Idziaszek, Wim Martens und Frank Neven (2010). „Simplifying XML schema: single-type approximations of regular tree languages“. In: *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data*. PODS '10. New York: ACM Press, S. 251–260.

Gelade, Wouter, Wim Martens und Frank Neven (2009). „Optimizing Schema Languages for XML: Numerical Constraints and Interleaving“. In: *SIAM Journal on Computing* 38.5, S. 2021–2043.

Ginsburg, Seymour und Michael A. Harrison (Apr. 1967). „Bracketed context-free languages“. In: *Journal of Computer and System Science* 1.1, S. 1–23. DOI: 10.1016/S0022-0000(67)80003-5.

Giordano, Richard (1995). „The TEI header and the documentation of electronic texts“. In: *Computers and the Humanities* 29 (1), S. 75–84. DOI: 10.1007/BF01830317.

– (1998). „The TEI Header and the Documentation of Electronic Texts“. In: *Text Encoding Initiative: Background and Context*. Kluwer Academic Publishers, S. 75–85.

Gippert, Jost (2006). „Linguistic documentation and the encoding of textual materials“. In: *Essentials of Language Documentation*. Hrsg. von Jost Gippert, Nikolaus P. Himmelmann und Ulrike Mosel. Bd. 178. Trends in Linguistics. Studies and Monographs. Berlin: Mouton de Gruyter, S. 337–361.

Gladky, A. V. und I. A. Melčuk (1969). „Tree grammars (Δ-grammars)“. In: *Proceedings of the 1969 conference on Computational linguistics*. COLING '69. Stroudsburg: Association for Computational Linguistics, S. 1–7. DOI: 10.3115/990403.990404.

Glushkov, Victor Mikhaylovich (1961). „The abstract Theory of Automata“. In: *Russian Mathematical Surveys (Uspekhi Matematicheskikh Nauk)* 16.5(101), S. 1–53. DOI: 10.1070/RM1961v016n05ABEH004112.

Goecke, Daniela, Harald Lüngen, Dieter Metzing, Maik Stührenberg und Andreas Witt (2010). „Different Views on Markup. Distinguishing Levels and Layers“. In: *Linguistic Modeling of Information and Markup Languages. Contributions to Language Technology*. Hrsg. von Andreas Witt und Dieter Metzing. Bd. 40. Text, Speech and Language Technology. Dordrecht: Springer, S. 1–22.

Goecke, Daniela, Maik Stührenberg und Andreas Witt (Mai 2008). „Influence of Text Type and Text Length on Anaphoric Annotation“. In: *Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis und Daniel Tapias. European Language Resources Association (ELRA). Marrakech.
→`http://www.lrec-conf.org/proceedings/lrec2008/pdf/368_paper.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Goldfarb, Charles F. (Mai 1973). *Design Considerations for Integrated Text Processing Systems*. Technical Report 320-2094. IBM Cambridge Scientific Center.
→`http://www.sgmlsource.com/history/G320-2094/G320-2094.htm` *[Letzter Abruf: 19. 04. 2012]*.
– (1978). *DCF GML User's Guide (IBM SH20-9160)*. IBM.
– (1981). „A Generalized Approach to Document Markup“. In: *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*. New York: ACM Press, S. 68–73.
– (1991a). „Introduction to Generalized Markup“. In: *The SGML Handbook*. Hrsg. von Yuri Rubinsky. Oxford: Oxford University Press. Kap. Annex A, S. 5–17.
– (1991b). *The SGML Handbook*. Hrsg. von Yuri Rubinsky. Oxford: Oxford University Press.
– (1996). *The Roots of SGML – A Personal Recollection*.
→`http://www.sgmlsource.com/history/roots.htm` *[Letzter Abruf: 19. 04. 2012]*.
– (1998). „Preface“. In: *Text Encoding Initiative: Background and Context*. 2. Aufl. Kluwer Academic Publishers, S. 1.

Goldfarb, Charles F., Edward J. Mosher und Theodore I. Peterson (1970). „An Online System for Integrated Text Processing“. In: *Proceedings of the 33rd Annual Meeting of the American Society for Information Science (ASIS Proceedings)*. Bd. 7.

Goldfarb, Charles F., Steven R. Newcomb, W. Eliot Kimber und Peter J. Newcomb (1997). *A Reader's Guide to the HyTime Standard*. Online.

Goldfarb, Charles F. und Paul Prescod (2004). *The XML Handbook*. 5. Aufl. The Charles F. Goldfarb Definite XML Series. Prentice Hall.

Good, Jeff (2002). *A Gentle Introduction to Metadata*.
→`http://www.language-archives.org/documents/gentle-intro.html` *[Letzter Abruf: 19.04.2012]*.

Goossens, Michel und Sebastian Rahtz (1999). *The LaTeXWeb Companion*. Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley Professional.

Gordon, Raymond G. Jr., Hrsg. (2005). *Ethnologue: Languages of the World*. 15. Aufl. Dallas: SIL International.

Götze, Michael und Stefanie Dipper (Apr. 2006). „ANNIS: Complex Multilevel Annotations in a Linguistic Database". In: *Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*. Hrsg. von David Ahn, Erik Tjong Kim Sang und Graham Wilcock. EACL. Trento, S. 61–64.

Gou, Gang und Rada Chirkova (2007). „Efficiently Querying Large XML Data Repositories: A Survey". In: *IEEE Transactions on Knowledge and Data Engineering* 19.10, S. 1381–1403.

Graf, John M. (Okt. 1988). „Ambiguity in the Instance". In: *TAG* 7, S. 6–9.

Graham, Ian S. und Liam R. E. Quin (1999). *XML Specification Guide*. Hrsg. von Cary Sullivan. Wiley Computer Publishing.

Grimes, Joseph E. (1974). *Word Lists and Languages*. Technical Report to the National Science Foundation 2. Ithaca: Department of Modern Languages und Linguistics, Cornell University.
→`http://www.eric.ed.gov/PDFS/ED100147.pdf` *[Letzter Abruf: 19.04.2012]*.

Gudgin, Martin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmakar und Yves Lafon (Apr. 2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C).
→`http://www.w3.org/TR/2007/REC-soap12-part1-20070427/` *[Letzter Abruf: 19.04.2012]*.

Habert, Benoît, Natalia Grabar, Pierre Jacquemart und Pierre Zweigenbaum (2001). „Building a text corpus for representing the variety of medical language". In: *Proceedings of the Corpus Linguistics 2001 conference*. Hrsg. von Paul Rayson, Andrew Wilson, Tony McEnery, Andrew Hardie und Shereen Khoja. UCREL Technical Paper 13.

Hadley, Marc (2002). *What's New in SOAP 1.2*.
→`http://hadleynet.org/marc/whatsnew.html` *[Letzter Abruf: 19.04.2012]*.

Harold, Elliotte Rusty (1999). *XML Bible*. Hungry Minds.
– (Sep. 2003). *An introduction to StAX*.
→`http://www.xml.com/pub/a/2003/09/17/stax.html` *[Letzter Abruf: 19.04.2012]*.
– (2009). *XOM*.

→http://www.cafeconleche.org/XOM/ *[Letzter Abruf: 19. 04. 2012]*.

Haupt, Stefanie und Maik Stührenberg (Aug. 2010). „Automatic upconversion using XSLT 2.0 and XProc“. In: *Proceedings of Balisage: The Markup Conference*. Bd. 5. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol5.Haupt01.

Hayashi, Yoshihiko, Thierry Declerck und Chiharu Narawa (Mai 2010). „LAF/ GrAF-grounded Representation of Dependency Structures“. In: *Language Resources: From Storyboard to Sustainability and LR Lifecycle Management, Workshop held at the seventh conference on International Language Resources and Evaluation (LREC 2010)*. Hrsg. von Victoria Arranz und Laura van Eerten. Valletta.

Heid, Ulrich, Helmut Schmidt, Kerstin Eckart und Erhard Hinrichs (Mai 2010). „A Corpus Representation Format for Linguistic Web Services: The D-SPIN Text Corpus Format and its Relationship with ISO Standards“. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner und Daniel Tapias. Valletta: European Language Resources Association (ELRA), S. 494–499.
→http://www.lrec-conf.org/proceedings/lrec2010/pdf/503_Paper.pdf *[Letzter Abruf: 19. 04. 2012]*.

Hilbert, Mirco, Oliver Schonefeld und Andreas Witt (Aug. 2005). „Making CONCUR work“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2005/Witt01/EML2005Witt01.xml *[Letzter Abruf: 19. 04. 2012]*.

Himmelmann, Nikolaus P. (Nov. 1998). „Documentary and descriptive linguistics“. In: *Linguistics* 39.1, S. 161–196. DOI: 10.1515/ling.1998.36.1.161.

– (2006). „Language documentation: What is it and what is it good for?“ In: *Essentials of Language Documentation*. Hrsg. von Jost Gippert, Nikolaus P. Himmelmann und Ulrike Mosel. Bd. 178. Trends in Linguistics. Studies and Monographs. Berlin: Mouton de Gruyter, S. 1–30.

Hopcroft, John E., Rajeev Motwani und Jeffrey D. Ullman (2000). *Introduction to Automata Theory, Languages, and Computation*. 2. Aufl. Amsterdam: Addison Wesley Longman, S. 521.

Hopcroft, John E. und Jeffrey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Hosoya, Haruo (2010). *Foundations of XML Processing. The Tree-Automata Approach*. Cambridge University Press.

Hosoya, Haruo und Makoto Murata (Okt. 2002). „Validation and boolean operations of attribute-element constraints“. In: *PLAN-X: Programming Languages Technologies for XML. Workshop colocated with PLI/ICFP/PPDG/GCSE-SAIG*. Pittsburgh.
→http://homepages.inf.ed.ac.uk/wadler/planx/planx-eproceed/papers/E00-1736272284.ps *[Letzter Abruf: 19. 04. 2012]*.

Huitfeldt, Claus (1999). *MECS – A multi-element code system*. Working papers of the Wittgenstein Archives at the University of Bergen. Wittgenstein Archives at the University of Bergen. Bergen.

Huitfeldt, Claus und C. M. Sperberg-McQueen (Feb. 2001). *TexMECS: An experimental markup meta-language for complex documents*. Forschungsbericht. Markup Languages und Complex Documents (MLCD) Project.
→`http://xml.coverpages.org/MLCD-texmecs20010510.html` *[Letzter Abruf: 19. 04. 2012]*.

– (Dez. 2004). „Markup Languages and Complex Documents (MLDC)“. Präsentation gehalten auf einer Kolloqiumsveranstaltung des Instituts der HKI (Historisch Kulturwissenschaftliche Informationsverarbeitung) der Universität zu Köln, 10. Dezember 2004.
→`http://old.hki.uni-koeln.de/material/huitfeldt.ppt` *[Letzter Abruf: 19. 04. 2012]*.

– (2006). „Representing and processing of GODDAG structures: implementation strategies and progress report“. In: *Proceedings of Extreme Markup Languages*. Montréal.

– (Juli 2010). „The MLCD Overlap Corpus (MOC)“. In: *Digital Humanities 2010 Conference Abstracts*. The Alliance of Digital Humanities Organisations u. a. London, S. 313–316.
→`http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/pdf/ab-633.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Iacob, Ionut Emil und Alex Dekhtyar (2004). „Parsing concurrent XML“. In: *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*. New York: ACM Press, S. 23–30.

– (2005a). „A framework for processing complex document-centric XML with overlapping structures“. In: *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. New York: ACM Press, S. 897–899.

– (Juni 2005b). „Towards a Query Language for Multihierarchical XML: Revisiting XPath“. In: *Proceedings of the 8th International Workshop on the Web & Databases (WebDB 2005)*. Baltimore, S. 49–54.

Ide, Nancy M. (1998). „Corpus Encoding Standard: SGML Guidelines for Encoding Linguistic Corpora“. In: *Proceedings of the First International Language Resources and Evaluation (LREC 1998)*. European Language Resources Association (ELRA). Granada, S. 463–470.

– (2000). „The XML Framework and Its Implication for Corpus Access and Use“. In: *Proceedings of Data Architectures and Software Support for Large Corpora*, S. 28–32.

– (2007). „Annotation Science: From Theory to Practice and Use“. In: *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Data Structures for Linguistic Resources and Applications. Proceedings of the Biennial GLDV Conference 2007*. Hrsg. von Georg Rehm, Andreas Witt und Lothar Lemnitzer. Tübingen: Gunter Narr Verlag, S. 3–7.

– (2009). „The American National Corpus: Then, Now, and Tomorrow“. In: *Selected Proceedings of the 2008 HCSNet Workshop on Designing the Australian National Corpus: Mustering Languages*. Somerville: Cascadilla Proceedings Project, S. 108–113.

Ide, Nancy M., Patrice Bonhomme und Laurent Romary (Mai 2000). „XCES: An XML-based Encoding Standard for Linguistic Corpora“. In: *Proceedings of the Second International Language Resources and Evaluation (LREC 2000)*. European Language Resources Association (ELRA). Athen, S. 825–830.

Ide, Nancy M. und Harry Bunt (Juli 2010). „Anatomy of Annotation Schemes: Mapping to GrAF“. In: *Proceedings of the Fourth Linguistic Annotation Workshop*. Hrsg. von Nianwen Xue, Massimo Poesio, Nancy M. Ide und Adam Meyers. Uppsala: Association for Computational Linguistics, S. 247–255.
→`http://aclweb.org/anthology/W/W10/W10-18.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Ide, Nancy M., Adam Kilgarriff und Laurent Romary (2000). „A Formal Model of Dictionary Structure and Content“. In: *Proceedings of Euralex*. Stuttgart, S. 113–126.

Ide, Nancy M. und Laurent Romary (2001a). „A common framework for syntactic annotation“. In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. ACL '01. Stroudsburg: Association for Computational Linguistics, S. 306–313. DOI: 10.3115/1073012.1073052.

– (2001b). „Standards for Language Resources“. In: *Proceedings of the IRCS Workshop on Linguistic Databases*. Philadelphia, S. 141–149.

– (Mai 2004a). „A Registry of Standard Data Categories for Linguistic Annotation“. In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 135–139.

– (2004b). „International Standard for a Linguistic Annotation Framework“. In: *Journal of Natural Language Engineering* 10.3-4, S. 211–225.

– (2007). „Towards International Standards for Language Resources“. In: *Evaluation of Text and Speech Systems*. Hrsg. von Laila Dybkjær, Holmer Hemsen und Wolfgang Minker. Springer, S. 263–284.

Ide, Nancy M., Laurent Romary und Tomaž Erjavec (Nov. 2001). „A Common XML-based Framework for Syntactic Annotations“. In: *Proceedings of the 1st NLP and XML Workshop*. Tokyo.

Ide, Nancy M. und C. M. Sperberg-McQueen (1995). „The TEI: History, goals, and future“. In: *Computers and the Humanities* 29 (1), S. 5–15. DOI: 10.1007/BF01830313.

Ide, Nancy M. und Keith Suderman (Mai 2004). „The American National Corpus First Release“. In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 1681–1684.
→`http://www.cs.vassar.edu/~ide/papers/LREC2004-ANC.pdf` *[Letzter Abruf: 19. 04. 2012]*.

– (Mai 2006a). „Integrating Linguistic Resources: The American National Corpus Model“. In: *Proceedings of the Fifth International Language Resources and Evaluation (LREC 2006)*. European Language Resources Association (ELRA). Genua.

– (2006b). „Merging Layered Annotations“. In: *Proceedings of Merging and Layering Linguistic Information, Workshop held in conjunction with LREC 2006*. Hrsg. von Erhard Hinrichs, Nancy M. Ide, Martha Palmer und James Pustejovsky. Genua, S. 61–64.

Ide, Nancy M. und Keith Suderman (Juni 2007). „GrAF: A Graph-based Format for Linguistic Annotations“. In: *Proceedings of the Linguistic Annotation Workshop*. Prag: Association for Computational Linguistics, S. 1–8.

– (Juli 2010). „Bridging the Gaps: Interoperability for GrAF, GATE, and UIMA“. In: *Proceedings of the Third Linguistic Annotation Workshop*. Hrsg. von Nianwen Xue, Massimo Poesio, Nancy M. Ide und Adam Meyers. Uppsala: Association for Computational Linguistics.
→http://aclweb.org/anthology/W/W10/W10-18.pdf *[Letzter Abruf: 19. 04. 2012]*.

Ide, Nancy M., Keith Suderman und Brian Simms (Mai 2010). „ANC2Go: A Web Application for Customized Corpus Creation“. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner und Daniel Tapias. Valletta: European Language Resources Association (ELRA), S. 3520–3525.
→http://www.lrec-conf.org/proceedings/lrec2010/pdf/745_Paper.pdf *[Letzter Abruf: 19. 04. 2012]*.

Institut für Deutsche Sprache. *Codierung der Korpora im XCES-Format*.
→http://www.ids-mannheim.de/cosmas2/projekt/referenz/ces.html *[Letzter Abruf: 19. 04. 2012]*.

– *IDS-Textmodell: Unterschiede gegenüber XCES*.
→http://www.ids-mannheim.de/kl/projekte/korpora/idsxces.html *[Letzter Abruf: 19. 04. 2012]*.

International Organization for Standardization (ISO) (2011). *International harmonized stage codes*.
→http://www.iso.org/iso/stage_codes.pdf *[Letzter Abruf: 19. 04. 2012]*.

ISO/IEC (2011a). *ISO/IEC Directives, Part 1: Procedures for the technical work*. 8. Aufl. International Organization for Standardization/International Electrotechnical Commission.
→http://isotc.iso.org/livelink/livelink?func=ll&objId=10563026&objAction=Open&nexturl=%2Flivelink%2Flivelink%3Ffunc%3Dll%26objId%3D4230455%26objAction%3Dbrowse%26sort%3Dsubtype *[Letzter Abruf: 19. 04. 2012]*.

– (2011b). *ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards*. 6. Aufl. International Organization for Standardization/International Electrotechnical Commission.
→http://isotc.iso.org/livelink/livelink?func=ll&objId=10562502&objAction=Open&nexturl=%2Flivelink%2Flivelink%3Ffunc%3Dll%26objId%3D4230456%26objAction%3Dbrowse%26sort%3Dsubtype *[Letzter Abruf: 19. 04. 2012]*.

Jacobs, Ian (März 2005a). *Über das W3C: Aktivitäten*. Übers. von Thomas Tikwinski.
→http://www.w3c.de/about/activities.html *[Letzter Abruf: 19. 04. 2012]*.

– (Okt. 2005b). *World Wide Web Consortium Process Document*.
→http://www.w3.org/2005/10/Process-20051014/ *[Letzter Abruf: 19. 04. 2012]*.

– (Juni 2007a). *About the World Wide Web Consortium (W3C)*.
→http://www.w3.org/Consortium/Overview *[Letzter Abruf: 19. 04. 2012]*.

– (Nov. 2007b). *About W3C: Groups*.
→http://www.w3.org/Consortium/activities *[Letzter Abruf: 19. 04. 2012]*.
– (Juni 2007c). *W3C Mission*.
→http://www.w3.org/Consortium/mission *[Letzter Abruf: 19. 04. 2012]*.

Jagadish, H. V., Laks V. S. Lakshmanan, Monica Scannapieco, Divesh Srivastava und Nuwee Wiwatwattana (Juni 2004). „Colorful XML: One Hierarchy Isn't Enough". In: *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 2004)*. New York: ACM Press, S. 251–262.

Jannidis, Fotis (2009). „TEI in a crystal ball". In: *Literary and Linguistic Computing* 24.3, S. 253–265. DOI: 10.1093/llc/fqp015.

Jelliffe, Rick (2001). *The Current State of the Art of Schema Languages for XML*. Presentation given at the XML Asia Pacific 2001 Conference. Sydney.
→http://www.estebel.com/articles/RickJelliffe.pdf *[Letzter Abruf: 19. 04. 2012]*.
– (Feb. 2002). *Resource Directory (RDDL) for Hook 0.2. A One-Element Language for Validation of XML Documents based on Partial Order*.
→http://xml.ascc.net/hook/ *[Letzter Abruf: 19. 04. 2012]*.
– (Jan. 2009). *Is Schematron a rules language?*
→http://broadcast.oreilly.com/2009/01/is-schematron-a-rules-language.html *[Letzter Abruf: 19. 04. 2012]*.
– (Okt. 2010). *Under-estimating XML as just a Tree. Composition, primary structure, internal links, external links*.
→http://broadcast.oreilly.com/2010/10/under-estimating-xml-as-just-a.html *[Letzter Abruf: 19. 04. 2012]*.

Jettka, Daniel (März 2011). „Repräsentation, Verarbeitung und Visualisierung multipler Hierarchien mit XStandoff und XSLT". Masterarbeit. Universität Bielefeld.
→http://www.daniel-jettka.de/pdf/Multiple_Hierarchien_mit_XStandoff_und_XSLT.pdf *[Letzter Abruf: 19. 04. 2012]*.

Jettka, Daniel und Maik Stührenberg (Aug. 2011). „Visualization of concurrent markup: From trees to graphs, from 2D to 3D". In: *Proceedings of Balisage: The Markup Conference*. Bd. 7. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol7.Jettka01.

Johansson, Stig (2008). „Some aspects of the development of corpus linguistics in the 1970s and 1980s". In: *Corpus Linguistics. An International Handbook*. Hrsg. von Anke Lüdeling und Merja Kytö. Bd. 29. Handbücher zur Sprach- und Kommunikationswissenschaft 2. Mouton de Gruyter, S. 33–53.

Kaplan, Ronald M. und Joan Bresnan (1982). „Lexical-Functional Grammar: A Formal System for Grammatical Representation". In: *The Mental Representation of Grammatical Relations*. Hrsg. von Joan Bresnan. Cambridge: MIT Press, S. 173–281.

Kawaguchi, Kohsuke (Mai 2001a). *Ambiguity Detection Of RELAX Grammars*.
→http://www.kohsuke.org/relaxng/ambiguity/AmbiguousGrammarDetection.pdf *[Letzter Abruf: 19. 04. 2012]*.
– (Mai 2001b). *W3C XML Schema: DOs and DON'Ts*.

→http://kohsuke.org/xmlschema/XMLSchemaDOsAndDONTs.html *[Letzter Abruf: 19. 04. 2012]*.

Kay, Michael (2004). *Up-conversion using XSLT 2.0*.
→http://www.saxonica.com/papers/ideadb-1.1/mhk-paper.xml *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2010a). „A Streaming XSLT Processor". In: *Proceedings of Balisage: The Markup Conference*. Bd. 5. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol5.Kay01.

Kilgarriff, Adam (Dez. 1999). *Generic Encoding Principles*. CONCEDE Deliverable.
→http://www.itri.brighton.ac.uk/projects/concede/DR2.1/generic.ps *[Letzter Abruf: 19. 04. 2012]*.

Kilpeläinen, Pekka (1999). „SGML and XML Content Models". In: *Markup Languages – Theory & Practice* 1.2, S. 53–76.

– (2011). „Checking determinism of XML Schema content models in optimal time". In: *Information Systems* 36.3. Special Issue on WISE 2009 – Web Information Systems Engineering, S. 596–617.

Kilpeläinen, Pekka und Rauno Tuhkanen (Juni 2007). „One-unambiguity of regular expressions with numeric occurrence indicators". In: *Information and Computation* 205.6, S. 890–916.

Kilpeläinen, Pekka und Derick Wood (2001). „SGML and XML Document Grammars and Exceptions". In: *Information and Computation* 169.2, S. 230–251.

Klarlund, Nils, Anders Møller und Michael I. Schwartzbach (1999). *Document Structure Description 1.0*. Techn. Ber. BRICS (Basic Research in Computer Science, Aarhus University).
→http://www.brics.dk/DSD/dsddoc.html *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2002). „The DSD Schema Language". In: *Automated Software Engineering* 9.3, S. 285–319. DOI: 10.1023/A:1016376608070.

Klarlund, Nils, Thomas Schwentick und Dan Suciu (2003). „XML: Model, Schemas, Types, Logics, and Queries". In: *Logics for Emerging Applications of Databases*. Hrsg. von Jan Chomicki, Ron van der Meyden und Gunter Saake. Berlin und Heidelberg: Springer, S. 1–41.

Kolb, Peter (2004). „Graphentheorie und Merkmalsstrukturen". In: *Computerlinguistik und Sprachtechnologie. Eine Einführung*. Hrsg. von Kai-Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde und Hagen Langer. 2. Aufl. München: Elsevier Spektrum Akademischer Verlag, S. 91–110.

Kosek, Jirka, Norman Walsh und Dick Hamilton (Mai 2006). *DocBook V5.0. The Transition Guide*.
→http://docbook.org/docs/howto/2006-05-16/ *[Letzter Abruf: 19. 04. 2012]*.

Kountz, Manuel, Ulrich Heid und Kerstin Eckart (Mai 2008). „A LAF/GrAF based Encoding Scheme for underspecified Representations of syntactic Annotations". In: *Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik,

Stelios Piperidis und Daniel Tapias. European Language Resources Association (ELRA). Marrakech.

Kracht, Marcus (2011). „Modal Logic Foundations of Markup Structures in Annotation Systems“. In: *Modeling, Learning and Processing of Text Technological Data Structures*. Hrsg. von Alexander Mehler, Kai-Uwe Kühnberger, Henning Lobin, Harald Lüngen, Angelika Storrer und Andreas Witt. Bd. 370. Studies in Computational Intelligence. Berlin und Heidelberg: Springer, S. 111–127.

Kupietz, Marc, Cyril Belica, Holger Keibel und Andreas Witt (Mai 2010). „The German Reference Corpus DeReKo: A Primordial Sample for Linguistic Research“. In: *Language Resources: From Storyboard to Sustainability and LR Lifecycle Management, Workshop held at the seventh conference on International Language Resources and Evaluation (LREC 2010)*. Hrsg. von Victoria Arranz und Laura van Eerten. Valletta, S. 1848–1854.

Kupietz, Marc, Oliver Schonefeld und Andreas Witt (Mai 2010). „The German Reference Corpus: New developments building on almost 50 years of experience“. In: *Language Resources: From Storyboard to Sustainability and LR Lifecycle Management, Workshop held at the seventh conference on International Language Resources and Evaluation (LREC 2010)*. Hrsg. von Victoria Arranz und Laura van Eerten. Valletta, S. 39–43.

Lämmel, Ralf, Stan Kitsis und Dave Remy (Nov. 2005). „Analysis of XML schema usage“. In: *Proceedings of the XML 2005 Conference*. Atlanta.

Langendoen, D. Terence und Gary F. Simons (1995). „A Rationale for the TEI Recommendations for Feature-Structure Markup“. In: *Computers and the Humanities* 29 (3), S. 191–209. DOI: 10.1007/BF01830616.

Larson, Martha, Valentin Jijkoun, Jobst Löffler und Erik Tjong Kim Sang (2007). „Practical applications of stand-off annotation“. In: *SDV – Sprache und Datenverarbeitung* 31.01/02, S. 115–129.

Lavagnino, John (2006). „When Not to Use TEI“. In: *Electronic Textual Editing*. Hrsg. von Lou Burnard, Katherine O'Brien O'Keeffe und John Unsworth. New York: The Modern Language Association of America, S. 334–338.

Layman, Andrew, Edward Jung, Eve Maler, Henry S. Thompson, Jean Paoli, John Tigue, Norbert H. Mikula und Steven J. DeRose (Jan. 1998). *XML-Data*. W3C Note. World Wide Web Consortium (W3C).
→`http://www.w3.org/TR/1998/NOTE-XML-data-0105/` *[Letzter Abruf: 19. 04. 2012]*.

Leech, Geoffrey (1993). „Corpus Annotation Schemes“. In: *Literary and Linguistic Computing* 8.4, S. 275–281.

– (2005). „Adding Linguistic Annotation“. In: *Developing Linguistic Corpora: a Guide to Good Practice*. Hrsg. von Martin Wynne. Oxford: Oxbow Books. Kap. 2, S. 17–29.

Lee, Dongwon und Wesley W. Chu (Sep. 2000). „Comparative Analysis of Six XML Schema Languages“. In: *ACM SIGMOD Record* 29.3, S. 76–87.

Lee, Dongwon, Murali Mani und Makoto Murata (Nov. 2000). *Reasoning about XML Schema Languages using Formal Language Theory*. Technical Report RJ# 10197, Log# 95071. IBM Almaden Research Center.

→http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.6406&rep=rep1&type=pdf *[Letzter Abruf: 19. 04. 2012]*.

Lee, Kiyong, Lou Burnard, Laurent Romary, Èric Villemonte de la Clergerie, Thierry Declerck, Syd Bauman, Harry Bunt, Lionel Clément, Tomaž Erjavec, Azim Roussanaly und Claude Roux (Mai 2004). „Towards an International Standard on Feature Structure Representation". In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 373–376.

Le Maitre, Jacques (2006). „Describing multistructured XML documents by means of delay nodes". In: *DocEng '06: Proceedings of the 2006 ACM symposium on Document engineering*. New York: ACM Press, S. 155–164.

Lemnitzer, Lothar, Laurent Romary und Andreas Witt (2012). „Representing human and machine dictionaries in Markup languages". In: *Dictionaries. An International Encyclopedia of Lexicography. Supplementary volume: Recent developments with special focus on computational lexicography*. Hrsg. von Rufus H. Gouws, Ulrich Heid, Wolfgang Schweickhard und Herbert Ernst Wiegand. Berlin und New York: Mouton de Gruyter.
→http://hal.inria.fr/inria-00441215/PDF/Lemnitzer_Romary_Witt-HSK-Article-v2009-12-15_-_deformatted.pdf.

Lemnitzer, Lothar und Heike Zinsmeister (2006). *Korpuslinguistik. Eine Einführung*. Narr Studienbücher. Tübingen: Gunter Narr Verlag.

Lewis, Emily (2009). *Microformats Made Simple*. New Riders.

Lezius, Wolfgang (Dez. 2002). „Ein Suchwerkzeug für syntaktisch annotierte Textkorpora". Diss. Institut für Maschinelle Sprachverarbeitung der Universität Stuttgart.

Lobin, Henning (2000). *Informationsmodellierung in XML und SGML*. Berlin und Heidelberg: Springer-Verlag.

Loiseau, Sylvain (Aug. 2007). „A formalism for representing milestone and hierarchical annotations together". In: *Proceedings of Extreme Markup Languages*. Montréal.

Lüngen, Harald und Mariana Hebborn (2010). „Konstruktionsgrammatische Analyse von Gliederungsstrukturen". In: *Semantic Approaches in Natural Language Processing. Proceedings of the 10th Conference on Natural Language Processing (KONVENS)*. Hrsg. von Manfred Pinkal, Ines Rehbein, Sabine Schulte im Walde und Angelika Storrer. Universitätsverlag des Saarlandes, S. 151–154.

Lüngen, Harald und Henning Lobin (Juni 2011). „Extracting domain knowledge from tables of contents". In: *Digital Humanities 2011 Conference Abstracts*. The Alliance of Digital Humanities Organisations u. a. Stanford University, S. 331–334.
→https://www.stanford.edu/group/dh2011/cgi-bin/wordpress/wp-content/uploads/2011/05/DH2011_BookOfAbs.pdf *[Letzter Abruf: 19. 04. 2012]*.

Lu, Shiyong, Yezhou Sun, Mustafa Atay und Farshad Fotouhi (Feb. 2005). „On the consistency of XML DTDs". In: *Data & Knowledge Engineering* 52.2, S. 231–247. DOI: 10.1016/j.datak.2004.05.007.

Maler, Eve und Jeanne El Andaloussi (1995). *Developing SGML DTDs: From Text to Model to Markup*. Upper Saddle River: Prentice Hall PTR.

Manguinhas, Hugo (Sep. 2009). „Schema Languages for XML“. In: *Actas do INForum - Simpósio de Informática 2009*. Hrsg. von Luis Rodrigues und Rui Lopes. Lissabon, Portugal: Faculdade de Ciências da Universidade de Lisboa, S. 299–310. →http://inforum.org.pt/INForum2009/docs/full/paper_35.pdf *[Letzter Abruf: 19. 04. 2012]*.

Mani, Murali (Aug. 2001). „Keeping chess alive: Do we need 1-unambiguous content models?“ In: *Proceedings of Extreme Markup Languages*. Montréal.

Marcoux, Yves (Aug. 2008a). „Graph characterization of overlap-only TexMECS and other overlapping markup formalisms“. In: *Proceedings of Balisage: The Markup Conference*. Bd. 1. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol1.Marcoux01.

– (Dez. 2008b). „Variants of GODDAGs and suitable first-layer semantics“. Presentation given at the Goddag workshop, Amsterdam, 1-5 December 2008.

Marcus, Mitchell P., Beatrice Santorini und Mary Ann Marcinkiewicz (1993). „Building a Large Annotated Corpus of English: The Penn Treebank“. In: *Computational Linguistics* 19.2, S. 313–330.

Marinelli, Paolo, Fabio Vitali und Stefano Zacchiroli (2008). „Towards the unification of formats for overlapping markup“. In: *New Review of Hypermedia and Multimedia* 14.1, S. 57–94.

Martens, Wim, Frank Neven und Thomas Schwentick (2005). „Which XML Schemas Admit 1-Pass Preorder Typing?“ In: *Database Theory – ICDT 2005*. Hrsg. von Thomas Eiter und Leonid Libkin. Bd. 3363. Lecture Notes in Computer Science. Berlin und Heidelberg: Springer, S. 68–82.

– (2009). „Complexity of Decision Problems for XML Schemas and Chain Regular Expressions“. In: *SIAM Journal on Computing* 39.4, S. 1486–1530.

Martens, Wim, Frank Neven, Thomas Schwentick und Geert Jan Bex (2006). „Expressiveness and Complexity of XML Schema“. In: *ACM Transactions on Database Systems (TODS)* 31.3, S. 770–813.

Maurice, Florence (2009). *Microformats – Semantik für Webseiten*. entwickler.press.

Maxwell, Michael (Mai 2010). „Standardization as a means to Sustainability“. In: *Language Resources: From Storyboard to Sustainability and LR Lifecycle Management, Workshop held at the seventh conference on International Language Resources and Evaluation (LREC 2010)*. Hrsg. von Victoria Arranz und Laura van Eerten. Valletta, S. 30–33.

McEnery, Anthony, Richard Xiao und Yukio Tono (2006a). *Corpus-Based Language Studies. An Advanced Resource Book*. Routledge Applied Linguistics. Oxon: Routledge.

– (2006b). „Corpus-Based Language Studies. An Advanced Resource Book“. In: Routledge Applied Linguistics. Oxon: Routledge. Kap. Corpus mark-up, S. 22–28.

McFadden, John und Sam Wilcott (März 1989). „Ambiguity in the Instance: An Analysis“. In: *TAG* 9, S. 3–5.

McGrath, Sean (2000). *XML Processing with Python*. The Charles F. Goldfarb Definite XML Series. Upper Saddle River: Prentice Hall PTR.

McKelvie, David, Amy Isard, Andreas Mengel, Morten Baun Møller, Michael Grosse und Marion Klein (2001). „The MATE workbench - An annotation tool for XML coded speech corpora". In: *Speech Communication. Special issue on "Speech Annotation and Corpus Tools"* 33.1–2, S. 97–112.

McLaughlin, Brett D. und Justin Edelson (März 2006). *Java and XML*. Sebastopol: O'Reilly Media.

Mehler, Alexander (2009). „Generalised Shortest Paths Trees: A Novel Graph Class Applied to Semiotic Networks". In: *Analysis of Complex Networks: From Biology to Linguistics*. Hrsg. von Matthias Dehmer und Frank Emmert-Streib. Weinheim: Wiley-VCH, S. 175–220.

Mengel, Andreas und Wolfgang Lezius (2000). „An XML-based encoding format for syntactically annotated corpora". In: *Proceedings of the Second International Conference on Language Resources and Engineering (LREC 2000)*. Athen, S. 21–126.

Meyer, Charles F. (2008). „Pre-electronic corpora". In: *Corpus Linguistics. An International Handbook*. Hrsg. von Anke Lüdeling und Merja Kytö. Bd. 29. Handbücher zur Sprach- und Kommunikationswissenschaft 2. Mouton de Gruyter, S. 1–14.

Michaelis, Jens und Uwe Mönnich (2007). „Towards a Logical Description of Trees in Annotation Graphs". In: *LDV Forum – Zeitschrift für Computerlinguistik und Sprachtechnologie* 22.2, S. 68–83.

Mintert, Stefan, Hrsg. (2002). *XML & Co*. edition W3C.de. Addison-Wesley.

Møller, Anders (Dez. 2002). *Document Structure Description 2.0*. Techn. Ber. BRICS (Basic Research in Computer Science, Aarhus University).
→`http://www.brics.dk/DSD/dsd2.html` *[Letzter Abruf: 19. 04. 2012]*.

Møller, Anders und Michael I. Schwartzbach (2006a). *An Introduction to XML and Web Technologies*. Harlow: Addison-Wesley.

– (2006b). „Schema Languages". In: *An Introduction to XML and Web Technologies*. Harlow: Addison-Wesley.

– (Jan. 2007). „XML Graphs in Program Analysis". In: *PEPM '07: Proceedings of the 2007 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation*. New York: ACM Press.

– (Juni 2011). „XML Graphs in Program Analysis". In: *Science of Computer Programming* 76.6, S. 492–515. DOI: 10.1016/j.scico.2009.11.007.

Müller, Stefan (2008). *Head-Driven Phrase Structure Grammar. Eine Einführung*. 2. Aufl. Bd. 17. Stauffenburg Einführungen. Tübingen: Stauffenburg Verlag.

Murata, Makoto (Mai 1995). *Forest-regular Languages and Tree-regular Languages*.
→`http://xml.coverpages.org/prelim1-980505-pdf.gz` *[Letzter Abruf: 19. 04. 2012]*.

– (Okt. 1999). *Hedge automata: a formal model for XML schemata*.
→`http://www.xml.gr.jp/relax/hedge_nice.html` *[Letzter Abruf: 19. 04. 2012]*.

Murata, Makoto und Haruo Hosoya (Aug. 2003). „Validation algorithm for attribute-

element constraints of RELAX NG“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2003/Murata01/EML2003Murata01.html *[Letzter Abruf: 19. 04. 2012]*.

Murata, Makoto, Dongwon Lee und Murali Mani (Aug. 2001). „Taxonomy of XML Schema Languages using Formal Language Theory“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2001/Murata01/EML2001Murata01.html *[Letzter Abruf: 19. 04. 2012]*.

Murata, Makoto, Dongwon Lee, Murali Mani und Kohsuke Kawaguchi (2005). „Taxonomy of XML Schema Languages Using Formal Language Theory“. In: *ACM Transactions on Internet Technology* 5.4, S. 660–704.

Mylonas, Elli und Allen H. Renear (1999). „The Text Encoding Initiative at 10: Not Just an Interchange Format Anymore — But a New Research Community“. In: *Computers and the Humanities* 33 (1), S. 1–9. DOI: 10.1023/A:1001832310939.

Nastase, Vivi und Stan Szpakowicz (2006). „Matching Syntactic-Semantic Graphs for Semantic Relation Assignment“. In: *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*. TextGraphs-1. Association for Computational Linguistics, S. 81–88.

Nelson, Theodor Holm (1982). *Literary Machines*. Mindful Press, Distributed by Eastgate Systems Inc.

– (Okt. 1997). *Embedded Markup Considered Harmful*.
→http://www.xml.com/pub/a/w3j/s3.nelson.html *[Letzter Abruf: 19. 04. 2012]*.

Nentwich, Christian (2005). *CLiX – A Validation Rule Language for XML*. Presented by Anthony Finkelstein at W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington D.C.
→http://www.w3.org/2004/12/rules-ws/paper/24/ *[Letzter Abruf: 19. 04. 2012]*.

Nentwich, Christian, Licia Capra, Wolfgang Emmerich und Anthony Finkelstein (2002). „xlinkit: A Consistency Checking and Smart Link Generation Service“. In: *Transactions on Internet Technology (TOIT)* 2.2, S. 151–185.
→http://www.cs.ucl.ac.uk/staff/a.finkelstein/papers/acmxlinkit.pdf *[Letzter Abruf: 19. 04. 2012]*.

Nentwich, Christian, Wolfgang Emmerich und Anthony Finkelstein (2001). *xlinkit: links that make sense*. Department of Computer Science, University College London.
→http://www.cs.ucl.ac.uk/staff/a.finkelstein/papers/htxlinkit.pdf *[Letzter Abruf: 19. 04. 2012]*.

Neven, Frank (2002). „Automata, Logic, and XML“. In: *Proceedings of the 16th International Workshop and 11th Annual Conference of the EACSL on Computer Science Logic*. Hrsg. von Julian C. Bradfield. CSL '02. London: Springer, S. 2–26.

Ng, Wilfred (2002). „Maintaining Consistency of Integrated XML Trees“. In: *Proceedings of the Third International Conference on Advances in Web-Age Information Management*. WAIM '02. London: Springer, S. 145–157.

Nicol, Gavin Thomas (Juni 2002a). *Attributed Range Algebra. Extending Core Range Algebra to Arbitrary Structures*.

– (Aug. 2002b). „Core Range Algebra: Toward a Formal Model of Markup“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2002/Nicol01/EML2002Nicol01.html` *[Letzter Abruf: 19. 04. 2012]*.

Obasanjo, Dare (Juli 2002). *W3C XML Schema Design Patterns: Dealing With Change*.
→`http://www.xml.com/pub/a/2002/07/03/schema_design.html` *[Letzter Abruf: 19. 04. 2012]*.

OMG (Aug. 2011a). *Unified Modeling Language – Infrastructure*. Object Management Group specification formal/2011-08-05. Object Management Group.
→`http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF` *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2011b). *Unified Modeling Language – Superstructure*. Object Management Group specification formal/2011-08-06. Object Management Group.
→`http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF` *[Letzter Abruf: 19. 04. 2012]*.

Papakonstantinou, Yannis und Victor Vianu (2000). „DTD inference for views of XML data“. In: *PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York: ACM Press, S. 35–46.

Pawson, Dave (Jan. 2005). *Docbook Frequently Asked Questions*.
→`http://www.dpawson.co.uk/docbook/` *[Letzter Abruf: 19. 04. 2012]*.

Pawson, Dave, Roger L. Costello und Florent Georges (Mai 2008). *ISO Schematron tutorial*.
→`http://www.dpawson.co.uk/schematron/` *[Letzter Abruf: 19. 04. 2012]*.

Pianta, Emanuele und Luisa Bentivogli (Mai 2004). „Annotating Discontinuous Structures in XML: the Multiword Case“. In: *Proceedings of LREC 2004 Workshop on "XML-based richly annotated corpora"*. Lisbon, Portugal, S. 30–37.

Piez, Wendell (2001). „Beyond the "descriptive vs. procedural" distinction“. In: *Markup Languages – Theory & Practice* 3.2, S. 141–172.

– (Aug. 2004). „Half-steps toward LMNL“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2004/Piez01/EML2004Piez01.html` *[Letzter Abruf: 19. 04. 2012]*.

– (Dez. 2008). „LMNL in Miniature“. Presentation given at the Goddag workshop, Amsterdam, 1-5 December 2008.
→`http://www.piez.org/wendell/LMNL/Amsterdam2008/presentation-slides.html` *[Letzter Abruf: 19. 04. 2012]*.

– (Juli 2010). „Towards Hermeneutic Markup: An architectural outline“. In: *Digital Humanities 2010 Conference Abstracts*. The Alliance of Digital Humanities Organisations u. a. London, S. 202–205.

→http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/pdf/ab-743.pdf *[Letzter Abruf: 19. 04. 2012]*.

Poesio, Massimo, Nils Diewald, Maik Stührenberg, Jon Chamberlain, Daniel Jettka, Daniela Goecke und Udo Kruschwitz (2011). „Markup Infrastructure for the Anaphoric Bank: Supporting Web Collaboration". In: *Modeling, Learning and Processing of Text Technological Data Structures*. Hrsg. von Alexander Mehler, Kai-Uwe Kühnberger, Henning Lobin, Harald Lüngen, Angelika Storrer und Andreas Witt. Bd. 370. Studies in Computational Intelligence. Berlin und Heidelberg: Springer, S. 175–195.

Pollard, Carl und Ivan A. Sag (1987). *Information-based Syntax and Semantics*. Menlo Park: CSLI, Center for the Study of Language and Information.

– (1994). *Head-Driven Phrase Structure Grammar*. Chicago: The University of Chicago Press.

Polyzotis, Neoklis und Minos Garofalakis (2002). „Statistical Synopses for Graph-Structured XML Databases". In: *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*. Hrsg. von Michael J. Franklin, Bongki Moon und Anastassia Ailamaki. ACM Press, S. 358–369. DOI: 10.1145/564691.564733.

Popescu-Belis, Andrei und Paula Estrella (2007). „Generating Usable Formats for Metadata and Annotations in a Large Meeting Corpus". In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prag: Association for Computational Linguistics.
→http://aclweb.org/anthology-new/P/P07/P07-2024.pdf.

Portier, Pierre-Edouard und Sylvie Calabretto (Aug. 2009). „Methodology for the construction of multi-structured documents". In: *Proceedings of Balisage: The Markup Conference*. Bd. 3. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol3.Portier01.

Powell, Gavin (2007). *Beginning XML Databases*. Indianapolis: Wiley Publishing.

Powers, Shelley (2003). *Practical RDF*. Sebastopol: O'Reilly.

Przepiórkowski, Adam (2004). *The IPI PAN Corpus: Preliminary version*. Instytut Podstaw Informatyki, Polska Akademia Nauk.
→http://nlp.ipipan.waw.pl/~adamp/Papers/2004-corpus/book_en.pdf *[Letzter Abruf: 19. 04. 2012]*.

Przepiórkowski, Adam und Piotr Bański (2009). „Which XML Standards for Multilevel Corpus Annotation?" In: *Proceedings of the 4th Language & Technology Conference*. Poznań, S. 245–250.

Przepiórkowski, Adam, Rafał L. Górski, Barbara Lewandowska-Tomaszczyk und Marek Łaziński (Mai 2008). „Towards the National Corpus of Polish". In: *Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008)*. Hrsg. von Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik,

Stelios Piperidis und Daniel Tapias. European Language Resources Association (ELRA). Marrakech.

Przepiórkowski, Adam, Zygmunt Krynicki, Łukasz Dębowski, Marcin Woliński, Daniel Janus und Piotr Bański (Mai 2004). „A Search Tool for Corpora with Positional Tagsets and Ambiguities“. In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 1235–1238.

Rabin, Michael Oser und Dana Stewart Scott (Apr. 1959). „Finite automata and their decision problems“. In: *IBM Journal of Research and Development* 3.2, S. 114–125.

Rahtz, Sebastian, Norman Walsh und Lou Burnard (2004). „A unified model for text markup: TEI, DocBook, and beyond“. In: *Proceedings of XML Europe 2004*.

Rehm, Georg, Oliver Schonefeld, Thorsten Trippel und Andreas Witt (Aug. 2010). „Sustainability of Linguistic Resources Revisited“. In: *Proceedings of the International Symposium on XML for the Long Haul: Issues in the Long-term Preservation of XML*. Bd. 6. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol6.Witt01.

Renear, Allen H. (1997). „Out of Praxis: Three (Meta) Theories of Textuality“. In: *Electronic Textuality: Investigations in Method and Theory*. Hrsg. von Kathryn Sutherland. Oxford: Clarendon Press, S. 107–126.

Renear, Allen H., Elli Mylonas und David G. Durand (Sep. 1996). „Refining our notion of what text really is: The problem of overlapping hierarchies“. In: *Selected Papers from the 1992 ALLC/ACH Conference, Christ Church, Oxford, April 1992*. Hrsg. von Nancy M. Ide und Susan Hockey. Bd. 4. Research in Humanities Computing. Oxford: Clarendon Press, S. 263–280.

Rizzi, Romeo (2001). „Complexity of Context-free Grammars with Exceptions and the inadequacy of grammars as models for XML and SGML“. In: *Markup Languages – Theory & Practice* 3.1, S. 107–116.

Robertson, Eddie (Juni 2002). *Combining Schematron with other XML Schema languages*. →`http://www.topologi.com/public/Schtrn_XSD/Paper.html` *[Letzter Abruf: 19. 04. 2012]*.

Rogers, James (Sep. 2003). „Syntactic Structures as Multi-dimensional Trees“. In: *Research on Language and Computation* 1.3-4, S. 265–305.

Romary, Laurent (2001). „An abstract model for the representation of multilingual terminological data: TMF – Terminological Markup Framework“. In: *Proceedings of TAMA 2001 – Sharing Terminological Knowledge. Antwerpen, 1.-2.2.2001*. Wien: Infoterm.

Romary, Laurent, Amir Zeldes und Florian Zipser (2011). „<tiger2/> – Serialising the ISO SynAF Syntactic Object Model“. In: *Computing Research Repository (CoRR)*. →`http://arxiv.org/abs/1108.0631` *[Letzter Abruf: 19. 04. 2012]*.

Rubinsky, Yuri und Murray Maloney (1997). *SGML on the Web: Small Steps Beyond HTML*. Charles F. Goldfarb Series On Open Information Management. Upper Saddle River: Prentice Hall.

Saake, Gunter und Kai-Uwe Sattler (2006). *Algorithmen und Datenstrukturen. Eine Einführung mit Java*. 3. überarbeitete Auflage. dpunkt.verlag.

Sasaki, Felix (Aug. 2004). „Secondary Information Structuring – A Methodology for the Vertical Interrelation of Information Ressources“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2004/Sasaki01/EML2004Sasaki01.html` *[Letzter Abruf: 19. 04. 2012]*.

– (März 2010). „How to avoid suffering from markup: A project report about the virtue of hiding XML“. In: *XML Prague 2010 Conference Proceedings*. Hrsg. von Jirka Kosek. ITI Series 2010-488. Institute for Theoretical Computer Science. Prag, S. 105–123.
→`http://www.xmlprague.cz/2010/files/XMLPrague_2010_Proceedings.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Sasaki, Felix und Andreas Witt (2004). „Linguistische Korpora“. In: *Texttechnologie. Perspektiven und Anwendungen*. Hrsg. von Henning Lobin und Lothar Lemnitzer. Tübingen: Stauffenberg Verlag, S. 195–216.

Schiller, Anne, Simone Teufel, Christine Stöckert und Christine Thielen (Aug. 1999). *Guidelines für das Taggen deutscher Textcorpora mit STTS*. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart/Seminar für Sprachwissenschaft, Universität Tübingen.
→`http://www.sfs.uni-tuebingen.de/resources/stts-1999.pdf` *[Letzter Abruf: 19. 04. 2012]*.

Schmidt, Thomas (2001). „The transcription system EXMARaLDA: An application of the annotation graph formalism as the Basis of a Database of Multilingual Spoken Discourse“. In: *Proceedings of the IRCS Workshop On Linguistic Databases, 11-13 December 2001*. Philadelphia: Institute for Research in Cognitive Science, University of Pennsylvania, S. 219–227.

Schmidt, Thomas, Christian Chiarcos, Timm Lehmberg, Georg Rehm, Andreas Witt und Erhard Hinrichs (2006). „Avoiding Data Graveyards: From Heterogeneous Data Collected in Multiple Research Projects to Sustainable Linguistic Resources“. In: *Proceedings of the EMELD'06 Workshop on Digital Language Documentation: Tools and Standards: The State of the Art*. Lansing, Michigan.

Schonefeld, Oliver (2007). „XCONCUR and XCONCUR-CL: A Constraint-Based Approach for the Validation of Concurrent Markup“. In: *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Data Structures for Linguistic Resources and Applications. Proceedings of the Biennial GLDV Conference 2007*. Hrsg. von Georg Rehm, Andreas Witt und Lothar Lemnitzer. Tübingen: Gunter Narr Verlag, S. 347–356.

– (Aug. 2008). „A Simple API for XCONCUR“. In: *Proceedings of Balisage: The Markup Conference*. Bd. 1. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol1.Schonefeld01.

Schonefeld, Oliver und Jan-Torsten Milde (Mai 2004). „Embedding IMDI Metadata into a Large Phonetic Corpus“. In: *Proceedings of the Fourth International Language Resources and Evaluation (LREC 2004)*. European Language Resources Association (ELRA). Lissabon, S. 623–626.

Schonefeld, Oliver und Andreas Witt (Aug. 2006). „Towards Validation of Concurrent Markup“. In: *Proceedings of Extreme Markup Languages*. Montréal.

Schraitle, Thomas (2004). *DocBook-XML. Medienneutrales und plattformunabhängiges Publizieren*. Nürnberg: SuSE Press.

Seemann, Michael (2003). *Native XML-Datenbanken im Praxiseinsatz*. Software & Support Verlag.

SGML Users' Group (Juni 1990). *A Brief History of the Development of SGML*.
→`http://www.sgmlsource.com/history/sgmlhist.htm` *[Letzter Abruf: 19. 04. 2012]*.

Simons, Gary F. (2007). „Linguistics as a community activity: The paradox of freedom through standards“. In: *Time and Again: Theoretical and Experimental Perspectives on Formal Linguistics. Papers in honor of D. Terence Langendoen*. Hrsg. von William D. Lewis, Simin Karimi, Heidi Harley und Scott Farrar. Amsterdam: John Benjamins.

Simons, Gary F. und Steven Bird (2003). „Building an Open Language Archives Community on the OAI Foundation“. In: *Library Hi Tech* 21.2, S. 12.

Sinclair, John (2005). „Corpus and Text – Basic Principles“. In: *Developing Linguistic Corpora: a Guide to Good Practice*. Hrsg. von Martin Wynne. Oxford: Oxbow Books. Kap. 1, S. 1–16.

Sperberg-McQueen, C. M. (Okt. 1992). *Poor-Folks SGML (PSGML). A Rational Subset of SGML*. TEI ED W 36/UIC CC DB 92-10.
→`http://www.tei-c.org/Vault/ED/edw36.gml` *[Letzter Abruf: 19. 04. 2012]*.

– (Apr. 2000). *Context-sensitive rules in XML Schema*.
→`http://www.w3.org/2000/04/26-csrules.html` *[Letzter Abruf: 19. 04. 2012]*.
– (Aug. 2002). „What matters?“ In: *Proceedings of Extreme Markup Languages*. Montréal.
– (Aug. 2003). „Logic grammars and XML Schema“. In: *Proceedings of Extreme Markup Languages*. Montréal.
– (2004). „Runways, product differentiation, snap-together joints, airplane glue, and switches that really switch“. In: *Proceedings of Extreme Markup Languages*. Montréal.
– (Aug. 2005). „Applications of Brzozowski derivatives to XML Schema processing“. In: *Proceedings of Extreme Markup Languages*. Montréal.
– (Aug. 2006). „Rabbit/Duck grammars: a validation method for overlapping structures“. In: *Proceedings of Extreme Markup Languages*. Montréal.
– (2007a). „Markup Languages and Schema Languages for Linguistic, Textual, Documentary Resources“. In: *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Data Structures for Linguistic Resources and Applications. Proceedings of the Biennial GLDV Conference 2007*. Hrsg. von Georg Rehm, Andreas Witt und Lothar Lemnitzer. Tübingen: Gunter Narr Verlag, S. 9–11.
– (Aug. 2007b). „Representation of overlapping structures“. In: *Proceedings of Extreme Markup Languages*. Montréal.
– (Okt. 2011). *TEI-P5-Dokumentgrammatik für das IDS-Textmodell*. Präsentation gehalten am 4. Oktober im IDS Mannheim.
→`http://www.blackmesatech.com/2011/10/ids/` *[Letzter Abruf: 19. 04. 2012]*.

Sperberg-McQueen, C. M. und Tim Bray (1997). „Extensible Markup Language (XML)“. In:

Proceedings of ACH-ALLC '97. Joint International Conference of the Association for Computers and the Humanities (ACH) and the Association for Literary & Linguistic Computing (ALLC). Queen's University. Kingston.
→http://xml.coverpages.org/sperbergXMLACH97.html *[Letzter Abruf: 19. 04. 2012]*.

„SGML Declarations for the TEI Guidelines" (Mai 1990). In: *TEI P1: Guidelines for Electronic Text Encoding and Interchange*. Hrsg. von C. M. Sperberg-McQueen und Lou Burnard. Oxford u. a.: published for the TEI Consortium by Humanities Computing Unit, University of Oxford. Kap. 3.2.

„Formal Grammar for the TEI-Interchange-Format Subset of SGML" (Nov. 1993a). In: *TEI P2: Guidelines for Electronic Text Encoding and Interchange*. Hrsg. von C. M. Sperberg-McQueen und Lou Burnard. Oxford u. a.: published for the TEI Consortium by Humanities Computing Unit, University of Oxford. Kap. 42.

Sperberg-McQueen, C. M. und Lou Burnard, Hrsg. (Nov. 1993b). *TEI P2: Guidelines for the Encoding and Interchange of Machine Readable Texts*. Oxford u. a.: published for the TEI Consortium by Humanities Computing Unit, University of Oxford.

– (1995). „The design of the TEI encoding scheme". In: *Computers and the Humanities* 29 (1), S. 17–39. DOI: 10.1007/BF01830314.

Sperberg-McQueen, C. M., David Dubin, Claus Huitfeldt und Allen H. Renear (Aug. 2002). „Drawing Inferences on the Basis of Markup". In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2002/CMSMcQ01/EML2002CMSMcQ01.html *[Letzter Abruf: 19. 04. 2012]*.

Sperberg-McQueen, C. M. und Claus Huitfeldt. *MOC-POC. MLCD Overlap Corpus (Proof of Concept)*.
→http://mlcd.blackmesatech.com/mlcd/2011/W/moc-poc/ *[Letzter Abruf: 19. 04. 2012]*.

– (1999a). „Concurrent Document Hierarchies in MECS and SGML". In: *Literary and Linguistic Computing* 14.1, S. 29–42.
→http://llc.oxfordjournals.org/content/14/1/29.full.pdf *[Letzter Abruf: 19. 04. 2012]*.

– (Juni 1999b). „GODDAG: A Data Structure for Overlapping Hierarchies". In: *Proceedings of ACH-ALLC1999. Joint International Conference of the Association for Computers and the Humanities (ACH) and the Association for Literary & Linguistic Computing (ALLC)*.
→http://www2.iath.virginia.edu/ach-allc.99/proceedings/sperberg-mcqueen.html *[Letzter Abruf: 19. 04. 2012]*.

– (2004). „GODDAG: A Data Structure for Overlapping Hierarchies". In: *Digital Documents: Systems and Principles, 8th International Conference on Digital Documents and Electronic Publishing, DDEP 2000, 5th International Workshop on the Principles of Digital Document Processing, PODDP 2000, Munich, Germany, September 13-15, 2000, Revised Papers*. Hrsg. von Peter King und Ethan V. Munson. Bd. 2023. Lecture Notes in Computer Science 2023. Springer, S. 139–160.

– (Dez. 2008a). „GODDAG". Presentation given at the Goddag workshop, Amsterdam, 1-5 December 2008.

– (Aug. 2008b). „Markup Discontinued Discontinuity in TexMecs, Goddag structures, and rabbit/duck grammars". In: *Proceedings of Balisage: The Markup Conference*. Bd. 1.

Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol1.Sperberg-McQueen01.

Sperberg-McQueen, C. M., Claus Huitfeldt und Allen H. Renear (2000). „Meaning and interpretation of markup". In: *Markup Languages – Theory & Practice* 2.3, S. 215–234.

Stede, Manfred und Heike Bieler (2011). „The MOTS Workbench". In: *Modeling, Learning and Processing of Text Technological Data Structures*. Hrsg. von Alexander Mehler, Kai-Uwe Kühnberger, Henning Lobin, Harald Lüngen, Angelika Storrer und Andreas Witt. Bd. 370. Studies in Computational Intelligence. Berlin und Heidelberg: Springer, S. 15–34.

Stegmann, Jens und Andreas Witt (Aug. 2009). „TEI Feature Structures as a Representation Format for Multiple Annotation and Generic XML Documents". In: *Proceedings of Balisage: The Markup Conference*. Bd. 3. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol3.Stegmann01.

Stührenberg, Maik (2008). „Sustainability of Text-Technological Resources". In: *Proceedings of the LREC 2008 Workshop "Sustainability of Language Resources and Tools for Natural Language Processing"*. Hrsg. von Andreas Witt, Georg Rehm, Thomas Schmidt, Khalid Choukri und Lou Burnard. ELRA/ELDA, S. 33–40.

Stührenberg, Maik und Daniela Goecke (Aug. 2008). „SGF – An integrated model for multiple annotations and its application in a linguistic domain". In: *Proceedings of Balisage: The Markup Conference*. Bd. 1. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol1.Stuehrenberg01.

Stührenberg, Maik, Daniela Goecke, Nils Diewald, Irene Cramer und Alexander Mehler (Juni 2007). „Web-based Annotation of Anaphoric Relations and Lexical Chains". In: *Proceedings of the Linguistic Annotation Workshop*. Hrsg. von Branimir Boguraev, Nancy M. Ide, Adam Meyers, Shigeko Nariyama, Manfred Stede, Janyce Wiebe und Graham Wilcock. Prag: Association for Computational Linguistics, S. 140–147.

Stührenberg, Maik und Daniel Jettka (Aug. 2009). „A toolkit for multi-dimensional markup: The development of SGF to XStandoff". In: *Proceedings of Balisage: The Markup Conference*. Bd. 3. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol3.Stuhrenberg01.

Stührenberg, Maik und Christian Wurm (Aug. 2010). „Refining the Taxonomy of XML Schema Languages. A new Approach for Categorizing XML Schema Languages in Terms of Processing Complexity". In: *Proceedings of Balisage: The Markup Conference*. Bd. 5. Balisage Series on Markup Technologies. Montréal. DOI: 10.4242/BalisageVol5.Stuhrenberg01.

Suda, Brian (Sep. 2006). *Using Microformats*. O'Reilly Short Cuts. O'Reilly Media.

Suderman, Keith und Nancy M. Ide (Apr. 2006). „Layering and Merging Linguistic Annotations". In: *Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*. Hrsg. von David Ahn, Erik Tjong Kim Sang und Graham Wilcock. EACL. Trento, S. 89–92.

Summerfield, Mark (2010). *Programming in Python 3. A Complete Introduction to the Python Language*. 2. Aufl. Developer's Library Series. Addison-Wesley.

Tanenbaum, Andrew Stuart (2003a). „Die Sicherungsschicht". In: *Computernetzwerke*. Übers. von Doris Heidenberger. 4. Aufl. Pearson Studium. Kap. 3.

– (2003b). „Einleitung". In: *Computernetzwerke*. Übers. von Doris Heidenberger. 4. Aufl. Pearson Studium. Kap. 1.

TEI Workgroup on Standoff Markup (2003). *Stand-off Markup*. Working Paper. Text Encoding Initiative (TEI).
→`http://www.tei-c.org/Activities/Workgroups/SO/sow06.xml` *[Letzter Abruf: 19. 04. 2012]*.

Tennison, Jeni (Aug. 2002). „Layered Markup and Annotation Language (LMNL)". In: *Proceedings of Extreme Markup Languages*. Montréal.

– (Mai 2007). „Creole: Validating Overlapping Markup". In: *Proceedings of XTech 2007: The Ubiquitous Web Conference*. Paris, France.

The Library of Congress (Juli 2005). *METS: Überblick und Anleitung*.
→`http://www.loc.gov/standards/mets/METSOverview.v2_de.html` *[Letzter Abruf: 19. 04. 2012]*.

Thompson, Henry S. und David McKelvie (1997). „Hyperlink semantics for standoff markup of read-only documents". In: *Proceedings of SGML Europe '97: The next decade – Pushing the Envelope*. Barcelona, S. 227–229.

Thulasirman, K. und M. N. S. Swamy (1992). „Acyclic Directed Graphs". In: *Graphs: Theory and Algorithms*. Wiley-Interscience. Kap. 5.7, S. 118–119.

Ule, Tylman (2002). *DEREKO Linguistic Markup*. Techn. Ber. Seminar für Sprachwissenschaft, Universität Tübingen.

Ule, Tylman und Erhard Hinrichs (2004). „Linguistische Annotation". In: *Texttechnologie. Perspektiven und Anwendungen*. Hrsg. von Henning Lobin und Lothar Lemnitzer. 217–243. Tübingen: Stauffenberg Verlag.

Valiente, Gabriel (2002). *Algorithms on Trees and Graphs*. Berlin und Heidelberg: Springer.

Van der Vlist, Eric (Dez. 2001). *Comparing XML Schema Languages*.
→`http://www.xml.com/pub/a/2001/12/12/schemacompare.html` *[Letzter Abruf: 19. 04. 2012]*.

– (Feb. 2003a). *Examplotron*.
→`http://examplotron.org/0/7/` *[Letzter Abruf: 19. 04. 2012]*.

– (2003b). *RELAX NG*. Sebastopol: O'Reilly.

– (2003c). *XML Schema*. Übers. von Gisbert Selke. Köln: O'Reilly.

– (2004). „ISO DSDL Overview". In: *Proceedings of XML Europe*. Amsterdam.

– (2007). „XML Schema Languages". In: *Advanced XML Applications*. Boston: Thomson Course Technology, S. 37–90.

Van Herwijnen, Eric (1994). *Practical SGML*. 2. Aufl. Kluwer Academic Publishers.

Vitali, Fabio, Nicola Amorosi und Nicola Gessa (Aug. 2003). „Datatype- and namespace-aware DTDs: A minimal extension". In: *Proceedings of Extreme Markup Languages*. Montréal.

→http://conferences.idealliance.org/extreme/html/2003/Gessa01/EML2003Gessa01.html *[Letzter Abruf: 19. 04. 2012]*.

Wahrig, Gerhard (1996). *Deutsches Wörterbuch*. Hrsg. von Renate Wahrig-Burfeind. Bertelsmann Lexikon Verlag.

Walmsley, Priscilla (2002). *Definitive XML Schema*. The Charles F. Goldfarb Definite XML Series. Prentice Hall PTR.

Walsh, Norman (Juli 1999). *Understanding XML Schemas*.
→http://www.xml.com/pub/a/1999/07/schemas/index.html *[Letzter Abruf: 19. 04. 2012]*.

Walsh, Norman und Leonard Muellner (1999). *DocBook: The Definitive Guide*. O'Reilly.

Welty, Christopher und Nancy M. Ide (1999). „Using the Right Tools: Enhancing retrieval From Marked-Up Documents". In: *Computers and the Humanities* 33, S. 59–84.

Widdows, Dominic und Beate Dorow (Aug. 2002). „A Graph Model for Unsupervised Lexical Acquisition". In: *Proceedings of the 19th International Conference on Computational Linguistics (COLING 19)*. Taipei, S. 1093–1099.

Widlöcher, Antoine und Yann Mathet (Juni 2009). „La plate-forme Glozz : environnement d'annotation et d'exploration de corpus". In: *Actes de la 16e conférence sur le Traitement Automatique des Langues Naturelles (TALN 2009) – Session posters*. Hrsg. von Mathieu Lafourcade und Violaine Prince. Association pour le Traitement Automatique des langues (ATALA). Senlis.

Wilde, Erik (Dez. 2002). „Making the Infoset Extensible". In: *Proceedings of XML 2002 conference*. Baltimore.
→http://dret.net/netdret/docs/wilde-xml2002.pdf *[Letzter Abruf: 19. 04. 2012]*.

– (2005). „Towards Conceptual Modeling for XML". In: *Proceedings of Berliner XML Tage 2005 (BXML 2005)*. Belin, S. 213–224.

– (Juni 2006a). „Structuring Content with XML". In: *Proceedings of the 10th International Conference on Electronic Publishing (ELPUB 2006)*. Bansko, Bulgaria.

– (Mai 2006b). „Tables and Trees Don't Mix (very well)". In: *Proceedings of the 15th International World Wide Web Conference (WWW2006)*. Edinburgh, S. 885–886. DOI: 10.1145/1135777.1135927.

Wilde, Erik und Robert J. Glushko (Juli 2008). „XML Fever". In: *Communications of the ACM* 51.7, S. 40–46. DOI: 10.1145/1364782.1364795.

Witt, Andreas (2002a). „Meaning and interpretation of concurrent markup". In: *Proceedings of ALLC-ACH2002, Joint Conference of the ALLC and ACH*. Tübingen.

– (2002b). „Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie". Diss. Universität Bielefeld.
→http://bieson.ub.uni-bielefeld.de/volltexte/2003/402/ *[Letzter Abruf: 19. 04. 2012]*.

– (Aug. 2004). „Multiple Hierarchies: New Aspects of an Old Solution". In: *Proceedings of Extreme Markup Languages*. Montréal.
→http://conferences.idealliance.org/extreme/html/2004/Witt01/EML2004Witt01.html *[Letzter Abruf: 19. 04. 2012]*.

Witt, Andreas, Daniela Goecke, Felix Sasaki und Harald Lüngen (2005). „Unification of XML Documents with Concurrent Markup“. In: *Literary and Linguistic Computing* 20.1, S. 103–116.

Witt, Andreas, Daniela Goecke, Maik Stührenberg und Dieter Metzing (2011). „Integrated Linguistic Annotation Models and their Application in the Domain of Antecedent Detection“. In: *Modeling, Learning and Processing of Text Technological Data Structures*. Hrsg. von Alexander Mehler, Kai-Uwe Kühnberger, Henning Lobin, Harald Lüngen, Angelika Storrer und Andreas Witt. Bd. 370. Studies in Computational Intelligence. Berlin und Heidelberg: Springer, S. 197–218.

Witt, Andreas, Ulrich Heid, Felix Sasaki und Gilles Sérasset (2009). „Multilingual language resources and interoperability“. In: *Language Resources and Evaluation* 43 (1), S. 1–14. DOI: 10.1007/s10579-009-9088-x.

Witt, Andreas, Georg Rehm, Erhard Hinrichs, Timm Lehmberg und Jens Stegmann (Juni 2009). „SusTEInability of Linguistic Resources through Feature Structures“. In: *Literary and Linguistic Computing* 24.3, S. 363–372.

Witt, Andreas, Oliver Schonefeld, Georg Rehm, Jonathan Khoo und Kilian Evang (Aug. 2007). „On the Lossless Transformation of Single-File, Multi-Layer Annotations into Multi-Rooted Trees“. In: *Proceedings of Extreme Markup Languages*. Montréal.
→`http://conferences.idealliance.org/extreme/html/2007/Witt01/EML2007Witt01.xml` *[Letzter Abruf: 19. 04. 2012]*.

Wittenburg, Peter, Daan Broeder und B. Sloman (Mai 2000). „Towards a Standard for Meta-Descriptions of Language Resources“. In: *Proceedings of the Second International Language Resources and Evaluation (LREC 2000)*. European Language Resources Association (ELRA). Athen.

Wittern, Christian, Arianna Ciula und Conal Tuohy (2009). „The making of TEI P5“. In: *Literary and Linguistic Computing* 24.3, S. 281–296. DOI: 10.1093/llc/fqp017.

Wörner, Kai (2009). „Werkzeuge zur flachen Annotation von Transkriptionen gesprochener Sprache“. Diss. Universität Bielefeld.
→`http://bieson.ub.uni-bielefeld.de/volltexte/2010/1669/pdf/diss_gold.pdf`.

Wynne, Martin (2001). „An archive for all of Europe: the TRACTOR initiative“. In: *Proceedings of the ACL/EACL Workshop on Sharing Tools and Resources for Research and Education*. Bd. 15. Stroudsburg: Association for Computational Linguistics, S. 59–62. DOI: 10.3115/1118062.1118073.

– (2005). „Archiving, Distribution and Preservation“. In: *Developing Linguistic Corpora: a Guide to Good Practice*. Hrsg. von Martin Wynne. Oxford: Oxbow Books. Kap. 6, S. 71–78.