Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie

Dissertation

zur Erlangung des akademischen Grades

Doctor philosophiae (Dr. phil.)

eingereicht an der

Fakultät für Linguistik und Literaturwissenschaft

– Universität Bielefeld –

von Andreas Witt

Gutachter: **Prof. Dr. Dieter Metzing (Universität Bielefeld)**

Prof. Dr. Henning Lobin (Justus-Liebig-Universität Gießen)

Inhalt

1 Einleitung

1.1 Ausgangspunkt

Seit der Veröffentlichung der ‚Standard Generalized Markup Language' durch die internationale Standardisierungsorganisation (ISO) im Jahr 1986 und insbesondere auch seit der Entwicklung und immens häufigen Verwendung der Hypertext Markup Language (HTML) ab dem Jahr 1992 wurden Auszeichnungssprachen (engl. Markup Languages) mehr und mehr auch zu einem Gegenstand der Informationswissenschaften und zunehmend auch der Linguistik. In dem mit der Sprachtechnologie befassten Bereich der Linguistik fiel diese Entwicklung mit einem – zumindest partiell vorzufindenden – Paradigmenwechsel zusammen.

Eine relativ traditionelle Schule der Computerlinguistik geht davon aus, dass der Beschreibungsgegenstand, d. h. die textuellen (oder im Prinzip auch auditiven) Sprachausschnitte, nicht nur streng von den Verarbeitungsschritten getrennt sind, sondern dass vielmehr die Sprachausschnitte auch nur so verwendet werden, wie sie in der Realität vorzufinden sind.

In neuerer Zeit finden jedoch immer häufiger Ansätze Anwendung, die unrestringierte Texte in verschiedenen Verarbeitungsstufen mit Wissen anreichern und dieses Wissen in die Ausgangsdaten inkorporieren. Derartig angereicherte Texte werden als Basis für weitere Verarbeitungsschritte verwendet. Ein solcher Ansatz wird z. B. von der Constraint Grammar (Karlson et al. 1995) gewählt, einem in starkem Maße an einer praktischen Verwendbarkeit orientierten Grammatikformalismus. Die Inkorporation des durch die Verarbeitung gewonnenen Wissens erfolgt in diesem Ansatz in einer Weise, die den Nutzern der Constraint Grammar verborgen bleibt: zwar werden die Texte sukzessiv mit Wissen angereichert, jedoch sind die Repräsentationen dieser Zwischenschritte im Normalfall nicht verfügbar, da sie nur intern als Grundlage für weitere sprachtechnologische Verarbeitungen dienen. Bei anderen Ansätzen werden jedoch die Zusatzinformationen explizit in die Texte eingefügt, ein Prozess, der als (linguistische) Annotation bezeichnet wird (vgl. Garside et al. 1997).

Der angesprochene Paradigmenwechsel ist darin zu sehen, dass die Sprachtechnologie in einem immer stärkeren Maße als Basis ihrer Verarbeitung nicht mehr die Texte verwendet, wie sie in der Realität vorkommen, sondern stattdessen auf annotierte Corpora zurückgreift. Dies kann als Rückschritt be-

zeichnet werden, da es zweifelsohne besser wäre, direkt die unrestringierten Texte als Basis der Sprachverarbeitung zu verwenden. Andererseits führt dieser Rückschritt zu einer deutlichen Verbesserung der Resultate der sprachtechnologischen Software, die auch als Lingware (*ling*uistic soft*ware*) bezeichnet wird. Der hier entwickelte Ansatz geht noch einen Schritt weiter: Die Annotationen werden auf verschiede Ebenen aufgeteilt, ein Schritt, der bewusst zum Einen eine Dominanz einer linguistischen Ebene vermeidet und sich zum Anderen von der Vorstellung löst, dass eine Integration verschiedener Beschreibungsebene in eine einzige Repräsentation möglich oder auch nur sinnvoll ist.

Das angesprochene zeitliche Zusammenfallen der zunehmenden Verwendung von Auszeichnungssprachen und der Verwendung linguistischer Annotation resultiert aus verschiedenen Faktoren, allerdings haben sich beide Entwicklungen wechselseitig beeinflusst, so dass die Innovation auf beiden Seiten beschleunigt wurde. Eine dieser positiven Wechselwirkungen zeigt sich darin, dass die linguistischen Annotationen immer häufiger in standardisierten Auszeichnungssprachen vorgenommen werden.

Den Ausgangspunkt der vorliegenden Untersuchung bildete der Wunsch, im Zusammenhang mit der Entstehung der Auszeichnungssprachen entwickelte Techniken für sprachtechnologische Anwendungen zu verwenden. Hierbei war in erster Linie daran gedacht, standardisierte linguistische Annotationen von Texten mit Auszeichnungssprachen als gegeben anzunehmen. Konkret fokussiert werden sollte hierbei die Fragestellung, inwieweit die den Auszeichnungssprachen inhärenten Möglichkeiten der Strukturüberprüfung, die normalerweise verwendet werden um den Aufbau von zu publizierenden Texten zu kontrollieren, ausgenutzt werden können, um sprachliche Ausdrucksmöglichkeiten zu beschränken.

Derartige Restriktionen der Sprache werden derzeit bereits in der Praxis verwendet, um bestimmte Textsorten (z. B. Technische Dokumentationen, Bedienungsanleitungen) in einem gegenüber der Standardsprache eingeschränkten linguistischen Inventar zu verfassen. Die Beschränkungen, die insbesondere das Lexikon und die Syntax betreffen, werden als Sprachkontrolle bezeichnet. Kontrollierte Sprachen sollen dazu beitragen, dass die potentiell in einem kollaborativen Prozess erstellten Texte kohärenter, verständlicher und maschinell einfacher verarbeitbar werden.

Die Verwendung von Auszeichnungssprachen für die Erstellung und Überprüfung Kontrollierter Sprachen erscheint als ein außerordentlich erstrebenswer-

tes Ziel, da in neuerer Zeit die textuellen Dokumente ohnehin in ‚Markup Languages' eingebettet sind und somit ein einziger Formalismus für verschiedene Zwecke verwendet würde. Bei der Annäherung an die Fragestellung, ob, wie und inwieweit dieses Ziel erreicht werden kann, wurde jedoch eine Reihe von grundlegenden Problemen sichtbar, die im Vorhinein geklärt werden mussten. Die Lösung dieser grundsätzlichen Probleme steht im Zentrum der vorliegenden Arbeit.

1.2 Begriffe und Konzepte

Der Titel der vorliegenden Arbeit lautet „Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie". Bereits hierin werden viele Begriffe eingeführt, die geklärt werden müssen.

Im Zentrum steht der Begriff der Informationsmodellierung oder genauer der Begriff der „textuellen Informationsmodellierung", wie er von Lobin (2000) beschrieben wird, nämlich als Bezeichnung für eine regelgeleitete Anordnung von textuellen Informationseinheiten. Lobin unterscheidet eine primäre und eine sekundäre Ebene der Informationsstrukturierung. Der Gegenstand der primären Ebene sind die textuellen Daten selbst sowie ihre Strukturierung, wohingegen die sekundäre Ebene beschreibt, wie die für die primären Ebenen verwendeten Regelwerke mit alternativen Regelwerken in Beziehung gesetzt werden können. Der Einteilung in eine primäre und eine sekundäre Informationsstrukturierung soll hier das Konzept der multiplen Informationsstrukturierung nebengeordnet werden. Dieses Konzept ist so zu verstehen, dass die primäre Ebene bei Bedarf vervielfacht wird – jedoch bezieht sich jede dieser Ebenen auf dieselbe Datengrundlage. Hierbei ergeben sich Auswirkungen auf die sekundäre Informationsstrukturierung.

Die Informationsmodellierung erfolgt mit Auszeichnungssprachen. Die angesprochene Standard Generalized Markup Language (SGML) stellt hierfür einen Rahmen dar, jedoch wurde dieser Formalismus seit seiner 1986 erfolgten Standardisierung nicht nur weiterentwickelt, sondern es wurde mit der Extensible Markup Language (XML) im Jahr 1998 eine wesentlich einfachere Untermenge dieser Sprache definiert, die zudem das derzeitige Zentrum weiterer Entwicklungen auf dem Gebiet der Auszeichnungssprachen darstellt.

In nahezu allen einführenden Texten zu SGML und XML wird darauf hingewiesen, dass das einzuhaltende Grundprinzip ihrer Verwendung in der Trennung von Form und Inhalt liegt. Dass dieses Grundprinzip nicht immer ein-

gehalten wird, zeigt sich aber z. B. bei der Betrachtung der Hypertext Markup Language. Textteile können mit Darstellungsinformationen (die Ebene der Form) und mit Strukturierungsinformationen (die Ebene des Inhalts) versehen werden - ein Zustand, der erst durch die Veröffentlichung einer sogenannten ‚strikten' HTML-Version partiell überwunden wurde. Es kann jedoch - trotz der Verstöße in der Praxis - davon gesprochen werden, dass das Ziel der Trennung von Inhalt und Form als Konsens anzusehen ist.

Auf das letzte anzusprechende Konzept wird mit dem Terminus ‚Sprachtechnologie' referiert. Die Sprachtechnologie beschäftigt sich mit der maschinellen Verarbeitung von Sprache, wobei das deutsche Wort Sprache als polysem anzusehen ist: zum Einen wird Sprache im Sinne von gesprochener Sprache verwendet, zum anderen jedoch in einem allgemeinen Sinn. Die englischsprachige Terminologie differenziert an diesem Punkt. In ihr wird der Begriff Speech Technology verwendet, wenn das automatische Erkennen bzw. die Generierung gesprochener Sprache fokussiert wird, während dagegen der Terminus Language Technology Aspekte der Sprachverarbeitung ins Zentrum rücken, die als unabhängig vom Modus der Präsentation, d. h. Schrift oder akustisches Signal, aufgefasst werden können. Bezogen auf die Ebenen linguistischer Beschreibungen ist die Speech Technology eher auf der Ebene der Phonetik, die Language Technology eher auf den Ebenen der Syntax und der Semantik angesiedelt. Der Nutzen des nachfolgend entwickelten Ansatzes ist insbesondere auf den letztgenannten Beschreibungseben zu sehen, d. h. der Begriff Sprachtechnologie wäre für die Erstellung einer englischen Version der vorliegenden Arbeit mit Language Technology zu übersetzen.

1.3 Die Problemstellung: Strukturierung linguistischer Information

Bei der Betrachtung der Erfordernisse an die Modellierung textueller Information fällt auf, dass die Strukturierung von sprachlichen Daten auf sehr unterschiedlichen linguistischen Ebenen erfolgen muss. Hierbei sind wesentlich detailliertere Ebenen als die Einteilung in Syntax und Semantik gemeint. Dies zeigt sich beispielsweise bei der Betrachtung der Regelwerke Kontrollierter Sprachen. Diese Regelwerke restringieren derartig unterschiedliche Aspekte textueller Daten, dass eine integrierte Modellierung zumindest nicht als sinnvoll erachtet werden kann.

Bei der Vorstellung des Konzeptes der Auszeichnungssprachen wurden bereits erwähnt, dass die Extensible Markup Language derzeit ein Zentrum

verschiedener Entwicklungen bildet und angedeutet, dass XML eine Vielzahl von Vorteilen besitzt. Aufgrund der Vorzüge gegenüber anderen Modellierungsformalismen sollte der entwickelte Ansatz auf der Extensible Markup Language basieren, wobei die weitergehenden Möglichkeiten von SGML selbstverständlich ebenfalls dargestellt und diskutiert werden.

Eine der Stärken von XML (und SGML) besteht darin, dass Informationen hierarchisch strukturiert werden. Dies erlaubt u. a. eine effiziente maschinelle Verarbeitung, allerdings ist diese Stärke zugleich eine der Schwächen. Informationen, die sich nicht in bestimmten Hierarchien (mittels mathematischer Bäume) strukturieren lassen, können nicht in einer natürlichen Weise in diesem Formalismus repräsentiert werden.

Eine Lösung dieses Problems liegt in der Aufteilung der Strukturierung auf verschiedene Ebenen. Diese Ebenenbildung ist jedoch nicht nur dann vorzunehmen, wenn die SGML-basierten Auszeichnungssprachen an ihre Grenzen stoßen. Vielmehr sollten in einem Informationsmodellierungsprozess die Ebenen unabhängig von diesen potentiellen Problemen herausgearbeitet werden. Die Ebenentrennung wird somit zu einer Frage der wissenschaftlichen Modell- bzw. Theoriebildung.

Eine Auszeichnung von Texten, die auf der Annahme unterschiedlicher Ebenen der Strukturierung basiert, stellt die Lösung für viele Modellierungsprobleme dar, allerdings gibt es bisher nur sehr wenige Ansätze, die diesen Weg beschreiten. Der Grund hierfür besteht zweifelsohne darin, dass die bisher vorgeschlagenen Herangehensweisen an diese Methodik sich nicht nur als sehr umständlich darstellen, sondern – um verwendbar zu sein – spezialisierte Software erfordern. Im Gegensatz dazu wird in der vorliegenden Arbeit ein Ansatz entwickelt, der es erlaubt textuelle Informationen bezüglich multipler Ebenen in einer direkten Weise zu modellieren. Dieser hier entwickelte Ansatz erlaubt nicht nur die angesprochenen Probleme zu lösen, sondern eröffnet völlig neuartige Perspektiven der Informationsstrukturierung.

1.4 Gliederung der Arbeit

In der vorliegenden Arbeit wird ein Annotationssystem entwickelt, das es erlaubt, verschiedene Ebenen der Informationsstrukturierung in eine Repräsentation zu integrieren. Hierfür werden neuere, standardisierte Auszeichnungssprachen verwendet.

Im Anschluss an die Einleitung wird die Thematik Auszeichnungssprachen prägnant vorgestellt (Kapitel 2). Der Titel des Kapitels – SGML und XML – zeigt bereits an, welche Auszeichnungssprachen thematisiert werden. In diesem Abschnitt werden alle Konzepte und Möglichkeiten der Standard Generalized Markup Language und der Extensible Markup Language angesprochen, die im weiteren Verlauf des Textes verwendet oder diskutiert werden. Der mögliche Umkehrschluss, dass die Darstellung alle Aspekte dieser Auszeichnungssprachen auslässt, die im Zusammenhang mit dieser Arbeit nicht relevant sind, gilt ebenso.

Fragen der mehrfachen Informationsstrukturierung mittels Auszeichnungssprachen werden im Kapitel 3 „Multiple Annotationsebenen“ diskutiert. Bisher verwendete oder vorgeschlagene Lösungen werden unter Hervorhebung ihrer Vor- und Nachteile angesprochen. All diese Ansätze stellen das Inventar dar, auf das das Annotationssystem zurückgreifen kann. Es werden einige der Möglichkeiten favorisiert, deren Manko jedoch darin liegt, dass sie kein formales Fundament besitzen.

In der Linguistik verwendete Formalisierungen, Merkmalsstrukturen, ihre logischen Beschreibungen und die mathematischen Operationen werden im darauffolgenden Kapitel, genauer im ersten Abschnitt dieses Kapitels angesprochen (Abschnitt 4.1 „Unifikationsbasierte Grammatiken“). Der dargestellte formale Apparat orientiert sich an den Ausarbeitungen von Bob Carpenter (1992), der in der Linguistik insbesondere durch seine Verwendung in der Grammatiktheorie HPSG (Pollard und Sag, 1994) bekannt wurde. Es wird im weiteren Verlauf dieses Kapitels gezeigt, wie in derartigen Formalismen repräsentierte Informationen in Auszeichnungssprachen ausgedrückt werden können. Hierbei wird auch auf Schwierigkeiten hingewiesen, auch die elaborierten Möglichkeiten derartiger Formalismen, insbesondere die Trennung von Modellierung und logischer Beschreibung, zu beachten.

Im Kapitel 5 wird eine „Architektur des Annotationssystems“ entwickelt. Dieses basiert auf XML (vgl. Kapitel 2) und erlaubt die Repräsentation multipler Annotationsebenen (vgl. Kapitel 3). Die im Abschnitt 5.2.2 vorgestellte

„Wissensrepräsentation annotierter Texte“ zur Darstellung von „Beziehungen zwischen Dokumenten“ bildet einen zentralen Teil der Formalisierung, ein Modell, welches einen Ansatz von Sperberg-McQueen, Huitfeldt und Renear (vgl. Abschnitt 5.2.1) erweitert und somit umfassender anwendbar werden lässt. Insbesondere erlaubt die Erweiterung die Betrachtung multipler Annotationsebenen. Der Abschnitt 5.2.2 zeigt dann, wie die im Kapitel 4 vorgestellten Techniken aus der Linguistik dazu verwendet werden können Beziehungen zwischen verschiedenen Ebenen (automatisch) herauszuarbeiten.

Das Kapitel 6 widmet sich der Problematik „Annotation linguistischer Informationen mit Markup-Sprachen“. Hierin werden zunächst verschiedene, bereits vorgeschlagene Annotationsschemata vorgestellt und kritisch hinterfragt. Anschließend werden einige Dokumentgrammatiken entwickelt, die es erlauben verschiedene linguistische Beschreibungsebenen zu annotieren. Sie besitzen einen exemplarischen Charakter, da das entwickelte Modell ein offenes Modell ist. Es soll keinesfalls impliziert werden, dass jedes dieser Annotationsschemata für die linguistische Auszeichnung notwendig ist oder gar, dass diese Menge von Dokumentgrammatiken vollständig ist.

Es ist bereits an der Gliederung der Arbeit zu erkennen, dass Informationsmodellierung im Fokus steht – ein eher theoretisches Thema. Dies ist zweifelsohne der Fall, jedoch soll diese Theorie kein Selbstzweck sein. Folglich widmet sich das die Arbeit abschließende Kapitel einer möglichen Anwendung – der Kontrollierten Sprache.

2 SGML und XML

2.1 Die Syntax von SGML

In diesem Abschnitt werden die für diese Arbeit fundamentalen Konstrukte aus SGML eingeführt. Dabei werden zum einen die Strukturierungskonstrukte ins Zentrum gestellt, andererseits werden die Möglichkeiten der Anpassung von SGML fokussiert, da dies einerseits die Mächtigkeit des Formalismus SGML begründet, gleichzeitig SGML jedoch zu einem sehr komplexen Apparat anwachsen ließ.

Ein SGML-Dokument besteht aus drei Teilen:

- einer SGML Deklaration,
- einer Definition der Dokumentenklasse und
- einer Dokumenteninstanz.

In der SGML Deklaration werden Grundeinstellungen vorgenommen, die auf alle anderen Komponenten von SGML Einfluss haben. In ihr werden die Bezeichnungen von Schlüsselwörtern, Möglichkeiten zum Weglassen bestimmter Annotationen, die Zeichenkodierung u. v. m. definiert. Viele der Konfigurationsmöglichkeiten von SGML, insbesondere die vielfältigen Möglichkeiten der Kapazitätsdefinitionen, sind durch die Weiterentwicklung der Hardware irrelevant geworden. Andere Möglichkeiten können durchaus auch nach wie vor als sinnvoll erachtet werden. Mittels der zweiten Komponente, der Definition der Dokumentenklasse, ist es möglich, eine allgemeine Struktur zu definieren, die verschiedenen Dokumenten zugewiesen werden kann. Hierfür wird eine sogenannte Document Type Definition (kurz DTD) erstellt bzw. es wird auf eine existierende DTD zurückgegriffen. Die DTD enthält eine Reihe von Element- und Attributdefinitionen. Die dritte Komponente, die Dokumentinstanz, beinhaltet das eigentliche Dokument, d. h. die textuellen Daten verbunden mit den in der DTD definierten Strukturinformationen.

2.1.1 SGML Deklaration

In der SGML Deklaration werden verschiedene Angaben über die Möglichkeiten der Verarbeitung von dem entsprechendem SGML Dokument vorgenommen. Im einzelnen sind diese Einstellungen folgendermaßen gruppiert:

Version: Am Beginn der SGML Deklaration befindet sich die Angabe der Version des SGML-Standards auf dem das SGML Dokument basiert. Von besonderer Bedeutung sind die Verweise auf den Originalstandard (durch “ISO 8879:1986“) und auf den Standard, der einen Anhang enthält[1], der SGML mit dem Ziel anpasste, den Standard besser für Intra- und Extranetze nutzen zu können. Der Verweis hierauf erfolgt mit der Zeichenkette “ISO 8879:1986 (WWW)“.

Zeichensatz: Die Angaben über den Zeichensatz bestehen aus zwei Komponenten: der Auswahl des Zeichensatzes (BASESET) und einer Interpretationsanweisung bezüglich des ausgewählten Zeichensatzes. Bis heute führt die Verwendung unterschiedlicher Zeichensätze durch verschiedene Programme oder unterschiedliche Dokumente zu einer Vielzahl von Problemen. SGML erlaubt bzw. erfordert es, den von den Dokumenten verwendeten Zeichensatz explizit zu machen. Damit kann ausgeschlossen werden, dass die SGML-Daten verarbeitenden Programme mit Heuristiken den Zeichensatz bestimmen müssen, ein Prozess bei dem Fehler auftreten können. Für die Zukunft ist jedoch eine allgemeine Akzeptanz des UNICODE-Standards zu erwarten. (vgl. ISO/IEC 10646) Dieser Zeichensatz ermöglicht nicht nur das Kodieren der lateinischen Buchstaben, sondern auch die Verwendung der meisten anderen Schriftsysteme - von Kyrillisch über Griechisch bis zu Chinesisch und vom phonetischen Alphabet IPA bis hin zu heute nicht mehr aktiv verwendeten Schriften (s. a. Gippert 1999).

Nachdem der Zeichensatz ausgewählt wurde, wird dieser genauer beschrieben, was meist dazu dient, bestimmte Zeichen des BASESET vom SGML-System anders zu interpretieren.

Kapazitäten: In dem Abschnitt CAPACITY werden Angaben über die maximale Größe des Gesamtdokumentes vorgenommen. Dies steht in der Tradition älterer Programmiersprachen, in denen z. B. neben dem Typ einer Variablen auch noch die maximale Speichernutzung ihrer möglichen Werte angegeben werden musste. Diese Möglichkeit der Selbstbeschränkung ist durch die heutige Rechnertechnologie redundant geworden. Konsequenterweise wurde in jüngerer Zeit die Möglichkeit geschaffen, die Kapazitätsbegrenzung durch die Angabe von „CAPACITY NONE“ auszuschalten.

[1] Es handelt sich um den im Jahr 1997 verabschiedeten Annex K „Web SGML Adaptations“.

Skopus: Im Abschnitt Skopus wird angegeben, ob sich die nachfolgend beschriebene Syntax auf das gesamte Dokument (einschließlich DTD) oder nur auf die Dokumenteninstanz bezieht.

Konkrete Syntax: SGML besitzt im ISO-Standard vordefinierte Syntaxvarianten. Die am häufigsten verwendete heißt Reference Concrete Syntax (RCS). Im Abschnitt SYNTAX wird angegeben, welche Syntaxvariante die Basis für das Dokument sein soll.

Darüber hinaus erlaubt dieser Abschnitt, die Reference Concrete Syntax zu verändern. Hierdurch ist es möglich, das Erscheinungsbild eines SGML Dokumentes beträchtlich zu verändern, die Unterschiede bestehen jedoch ausschließlich aus Ersetzungen von bestimmten Zeichen. Beispielsweise können die Markierungen von Umgebungen durch die Angabe von (1) `STAGO Beginn: ,`(2) `ETAGO Ende:` und (3) `TAGC !` so verändert werden, dass die normalerweise die Elementnamen umschließenden spitzen Klammern (<, </, >) durch – für deutschsprachige Menschen – explizitere Markierungen ersetzt werden.[2] Häufiger als derartige Redefinitionen von speziellen Zeichen finden die im Abschnitt SYNTAX vorzunehmenden Angaben zur Groß- und Kleinschreibung von Strukturinformationen Anwendung. So ist in HTML-Dokumenten diese Schreibweise irrelevant, d. h. es kann z. B. mit `<HtML>` beginnen und mit `</html>` enden. Durch eine Angabe von `NAMECASE GENERAL YES` wird erreicht, dass nur die exakte Schreibung erlaubt ist.

Verwendung spezieller Möglichkeiten: In der SGML Deklaration können im Abschnitt `FEATURE` spezielle Syntaxmöglichkeiten aktiviert werden. Diese umfassen Angaben über Auslassbarkeit von Strukturinformationen (`OMITTAG YES`), der Erlaubnis gesamte SGML-Dokumente als Teildokument einzubinden (`SUBDOC YES`), die Zulassung ein SGML Dokument mit mehr als einer DTD auszustatten (`CONCUR YES`) und die Möglichkeit der Verbindung zwischen verschiedenen Dokumenttypen (`LINK linktyp`).

Zu den speziellen Möglichkeiten gehört es, bestimmte inferierbare Markierungen auszulassen (`OMITTAG YES`). Von dieser Option wird häufig Gebrauch gemacht, z. B. in HTML-Dokumenten. Das Ziel dieser

[2] Dadurch wäre es z. B. möglich, ein HTML Dokument mit `Beginn: HTML!` anzufangen und mit `Ende: HTML!` zu beenden.

Option ist es, durch die Auslassung gewisser Strukturinformationen eine SGML-Instanz zu verkleinern, was einerseits dazu führt, weniger Speicherplatz zu benötigen, andererseits den Aufwand bei der textuellen Erfassung der Daten zu verringern. Mit denselben Zielen sind im Abschnitt `FEATURE` weitere spezielle Möglichkeiten vorgesehen, die äußerst selten angewendet wurden. Die Strukturierung von ähnlichen Elementen, die durch numerische Spezifizierung unterschieden werden, kann durch die Zuweisung von Rängen knapper definiert werden (`RANK YES`), bestimmte Strukturinformationen können verkürzt geschrieben werden (`SHORTTAG YES`) und gewisse textuelle Daten müssen nicht mit Strukturinformationen versehen werden, da Schablonen zum Einsatz kommen können (`DATATAG YES`).

Es ist nicht notwendig, die SGML-Deklaration anzugeben. Wenn auf ihre Angabe verzichtet wird, wird eine der bereits erwähnten vordefinierten Deklarationen angenommen, z. B. die RCS.

2.1.2 Dokumenttypdefinition

Im Gegensatz zur SGML-Deklaration kann auf die Angabe einer Document Type Definition nicht verzichtet werden, d. h. jedes SGML-Dokument muss eine DTD beinhalten. Eine DTD definiert die Struktur eines Dokumentes in einer allgemeinen Weise, was dazu führt, dass normalerweise mehr als eine Dokumentenausprägung für eine DTD existiert. Ähnlich wie natürliche Sprachen eine Grammatik besitzen, die Sätze oder Äußerungen einer Sprache legitimiert, bildet die DTD eine Grammatik für Dokumente.

Eine DTD beschreibt somit die Gemeinsamkeiten einer Klasse von Dokumenten. Die Beschreibung erfolgt in einer formalen Weise, wobei nur sogenannte Elemente und Attribute verwendet werden. Darüber hinaus stehen variablenähnliche Konstrukte zur Verfügung, die als Entitäten bezeichnet werden.

2.1.2.1 Elemente

Elemente dienen zur Kennzeichnung von Abschnitten bzw. Umgebungen in Dokumenten. Elemente erlauben es, die Umgebungen zu markieren, wofür dem Element ein Bezeichner zugewiesen wird. Diese Umgebung markiert einen Dokumententeil, der wiederum eine interne Struktur besitzen kann. Es ist selbstverständlich möglich, die Strukturen von Umgebungen auszudrü-

cken, wofür den Elementen Inhaltsmodelle zugewiesen werden. Der allgemeine Aufbau einer Elementdefinition sieht folgendermaßen aus:

```
<!ELEMENT Elementname Inhaltsmodell>
```

Die möglichen Inhaltsmodelle können folgendermaßen klassifiziert werden:

- Ein Element kann eine Umgebung markieren, in der sich nur textuell repräsentierte Daten befinden. Dies wird durch das Schlüsselwort `#PCDATA` zum Ausdruck gebracht.
- Ein Umgebung beinhaltet eine Kombination weiterer Elemente. Alle Elemente, die in der DTD definiert wurden, können im Inhaltsmodell auftreten.
- Mischformen von Daten (`#PCDATA`) und Elementen werden als Inhaltsmodell definiert.
- Eine Umgebung darf weder Text- noch Strukturinformationen beinhalten. Für diesen Fall wird durch die Angabe des Schlüsselwortes `EMPTY` ein leeres Inhaltsmodell definiert.
- Das Inhaltsmodell wird nicht restringiert. Das Schlüsselwortes `ANY` zeigt an, dass beliebige Dateninhalte in einer Umgebung vorkommen dürfen.

Die einfachsten Formen von Inhaltsmodellen bilden die durch `#PCDATA` und die durch `EMPTY` gekennzeichneten. Diese können als die atomaren Bausteine strukturierter Dokumente betrachtet werden. Wie kompliziert oder wie einfach auch die Struktur der Dokumente sein mag, Dateninhalte oder textuelle Inhalte müssen vorkommen, da sie die Träger der eigentlichen Information bilden.

Die Inhaltsmodelle, die weitere Elemente in sich versammeln, sind komplexer aufgebaut. Sie geben nicht nur die Namen, der in der durch sie definierten Umgebung zugelassenen Elemente an, sondern auch deren potentielle Abfolge und den Verpflichtungsgrad ihres Auftretens. Dies alles erfolgt durch die Operatoren Komma (`,`), vertikaler Trennstrich (`|`), Kaufmannsund (`&`), Plus (`+`), Fragezeichen (`?`) und Stern (`*`). Im einzelnen bilden das Komma und das Kaufmannsund die Operatoren, die die lineare Abfolge definieren. Die Verbindung zweier Elemente mit dem Zeichen ‚Komma' besagt, dass die Elemente in der Dokumenteninstanz in der Abfolge ihres Erscheinens in der DTD auftre-

ten müssen, wohingegen durch das Zeichen ‚&' angegeben wird, dass die Abfolge beider Umgebungen irrelevant ist. Das Plus, der Stern und das Fragezeichen geben an, ob und wie häufig Elemente auftreten müssen bzw. dürfen. Sie werden hinter den Elementbezeichner geschrieben, auf den sie sich beziehen. Das Fragezeichen (?) drückt aus, dass die durch sie spezifizierte Umgebung maximal einmal vorkommen darf. Wenn das Element hingegen mindestens einmal auftreten muss, wird dies durch ‚Plus' (+) gekennzeichnet. Der Stern (*) besagt, dass das betreffende Element unrestringiert häufig auftreten darf, d. h. insbesondere auch, dass es nicht auftreten muss. Kommt der Umgebungsbezeichner ohne weitere Spezifizierung in der DTD vor, muss das Element genau einmal im Dokument vorkommen. In gewisser Weise kann somit ein weiterer Operator aufgeführt werden: die leere Zeichenkette. Der vertikale Trennstrich lässt sich nicht eindeutig dem reihenfolge- oder dem auftretensspezifizierenden Operatoren zurechnen, da er besagt, dass eines der beiden in der DTD durch Trennstrich (|) verbundene Elemente auftreten muss.

Eine weitere Unterscheidung der Operatoren kann auf der syntaktischen Ebene vorgenommen werden:

- die Operatoren &, | und , sind zweistellig und
- das Plus, der Stern das Fragezeichen und die leere Zeichenkette sind monadisch.

Eine Gruppierung von Inhaltsmodellen ist möglich und erfolgt durch runde Klammern. Diese gruppierten Inhaltsmodelle können wiederum durch die zweistelligen Operatoren verbunden bzw. durch die monadischen Operatoren spezifiziert werden.

2.1.2.2 Attribute

Elemente ermöglichen prinzipiell eine eindeutige Auszeichnung des zu strukturierenden Textes in einem beliebigen Detailliertheitsgrad. Umgebungen, die eigene logische Einheiten markieren sollen, können durch unterschiedliche Namen gekennzeichnet werden. Das Elementreservoir kann also quasi unbegrenzt vergrößert werden. Jedoch erscheint es häufig sinnvoll, Ähnlichkeiten von Umgebungen zum Ausdruck zu bringen, was durch die bloße Vergrößerung der Elementmenge nicht möglich ist. Um derartige Modifikationen von

Umgebungen vorzunehmen, stellt SGML Attribute zur Verfügung. Sie werden in der DTD folgendermaßen definiert:

```
<!ATTLIST Elementname
(Attributname Wertebereich Verpflichtungsgrad/Default)+>
```

Der Elementname bezeichnet das Element, welches durch Attribute modifizierbar werden soll. Anschließend folgt – hier in Klammern – mindestens eine Attributdefinition. Jedes Attribut erhält einen frei wählbaren Namen. Hierzu wird ein Attributname vergeben. Durch die Angabe eines Wertebereichs wird die Auswahl der möglichen Werte beschränkt. Des Weiteren wird in der DTD angegeben, ob der Wert gesetzt werden muss (Schlüsselwort `#REQUIERED`) oder darf (`#IMPLIED`) und ob das Attribut bei nicht angegebenen Attributwerten einen vordefinierter Wert annehmen soll. Es ist auch möglich, vordefinierte und nicht veränderbare Werte anzugeben (`#FIXED festwert`).

Attributwerte können folgende Typen besitzen:

Zeichen: Das Schlüsselwort `CDATA` gibt an, dass der Wert des Attributes eine nicht weiter restringierte Zeichenkette ist.

Zahlen: Die Schlüsselwörter `NUMBER` und `NUMBERS` geben an, dass der Wert eine Zahl bzw. eine Kette mehrerer durch Leerzeichen voneinander getrennter Zahlen sein muss.

Auswahlliste: Die Angabe einer Auswahlliste erlaubt eine feste Vorgabe einer Liste möglicher Werte. Entsprechend des Verpflichtungsgrades kann oder muss ein Wert aus dieser Liste ausgewählt werden.

Entitäten: Durch die Schlüsselwörter `ENTITIES` und `ENTITY` kann der Wertebereich auf die Einheiten begrenzt werden, die separat – gewissermaßen als Variablen (vgl. Absatz 2.1.2.3) – definiert wurden.

Referenzpunkte: Durch die Angabe des Schlüsselwortes `ID` wird angegeben, dass der Wert des Attributes ein Identifikator sein muss. Dieser Identifikator darf in dem durch die DTD beschriebenen Dokument nur einmal auftreten.

Verweise: Wenn auf einen mit `ID` definierten Referenzpunkt direkt verwiesen werden soll, geschieht dies durch die Angabe des Wertebereichs `IDREF`. Durch das Schlüsselwort `IDREFS` kann mit dem Attributwert nicht nur auf einen Referenzpunkt verwiesen werden, sondern auch auf eine Kette von Identifikatoren.

In vielen Fällen ist es möglich, Informationen sowohl durch Attribute als auch durch Elemente auszudrücken. Es ist Aufgabe der DTD-Designer die sinnvollste Informationsstruktur auszuwählen.

2.1.2.3 Entitäten

Für den Zweck einer verbesserten Speicherung und für den Zugriff auf Daten werden in SGML Entitäten verwendet. Ähnliche Zwecke erfüllen in Programmiersprachen Variablen, wobei allerdings wichtige Unterschiede festzuhalten sind. Variablen können ihren Wert im Verlauf der Verarbeitung ändern, Entitäten nicht. Die Ursache hierfür ist insbesondere darin zu sehen, dass in SGML versucht wird, Informationen deklarativ zu repräsentieren. Eine Veränderung von Variablenwerten würde dem Prinzip zuwiderlaufen. Eine treffendere Analogie zu Konstrukten in Programmiersprachen ist deshalb die zu den in einigen, insbesondere älteren Programmiersprachen, existierenden Variablen, deren Werte nicht veränderbar sind und die Konstanten genannt werden. Derselben Entität können zwar verschiedene Werte zugewiesen werden, jedoch findet nur die erste Wertzuweisung Verwendung.

In SGML existieren zwei unterschiedliche Arten von Entitäten: Parameterentitäten und allgemeine Entitäten. Beide Typen werden in der DTD definiert, verwendet werden sie in unterschiedlichen Teilen eines SGML-Dokumentes: Parameterentitäten kommen in der DTD zum Einsatz, während allgemeine Entitäten in den Dokumenteninstanzen benutzt werden. Die allgemeine Form der Definition von Entitäten lautet:

```
allgemeine Entität:   <!ENTITY   Entitätenname Wert>
Parameterentität:     <!ENTITY % Entitätenname Wert>
```

Der Aufruf der Entitäten geschieht wie folgt[3]:

```
allgemeine Entität (in der Instanz):   &Entitätenname;
Parameterentität (in der DTD):          %Entitätenname;
```

Dem Einsatz von Parameterentitäten kommt bei der Entwicklung von DTDs eine große Bedeutung zu. Sie werden insbesondere zur Strukturierung von DTDs verwendet. Zum einen erlauben sie eine bessere Lesbarkeit, zum anderen ermöglichen sie in vielen Fällen aber erst eine Modifizierbarkeit der

[3] Der Verweis auf den Aufruf der allgemeinen Entitäten soll nur kontrastiven Zwecken dienen. Die Dokumentinstanz wird in Abschnitt 2.1.3 eingeführt.

DTDs, da durch ihren Einsatz die DTD-Designer in die Lage versetzt werden, Definitionen in einer konsistenten, die gesamte DTD betreffenden Form vorzunehmen bzw. zu ändern. Ein Einsatzgebiet von allgemeinen Entitäten ist ihre Verwendung als Platzhalter für Daten der Dokumentinstanz, die aus unterschiedlichen Gründen nicht direkt eingefügt werden sollen oder können. Gründe für die Wahl dieser Dateninklusionsmethode können u. a. darin liegen, dass die Zeichen nicht über Tastaturen eingegeben werden können (z. B. Ø, Δ, £), nichttextuelle Daten, z. B. Bilder eingefügt werden sollen oder dass Textbausteine wiederverwendet werden sollen. Ein weiteres wichtiges Einsatzgebiet von Entitäten findet sich in der physikalischen Strukturierung von SGML-Dokumenten. Sowohl Parameterentitäten als auch allgemeine Entitäten können auf Dateien verweisen, die Teile der DTD (Parameterentitäten) bzw. Teile der Dokumentinstanz, z. B. einzelne Abschnitte eines Textes, beinhalten. Dadurch ist es möglich, ein logisches SGML-Dokument, eine logische DTD oder eine logische SGML-Instanz auf verschiedene physikalische Dateien zu verteilen.

2.1.2.4 Notationen

Für die Referenz auf Daten können in der DTD Notationen definiert werden. Derartige Daten können einer völlig anderen Syntax unterliegen, z. B. kann durch Notationen eine Notation für Grafik- oder Audioformate festgelegt werden. Es gibt allerdings auch textbasierte Daten für die eigene Notationen erklärt werden, z. B.: im (La-)TeX-Format geschriebene Formeln oder auch Daten, die ihrerseits in SGML repräsentiert sind. In einer Notationsdefinition wird ein Name für eine Notation und deren Typ bzw. deren Syntax spezifiziert. Die allgemeine Form sieht wie folgt aus:

```
<!NOTATION Notationsname Notationstypfestlegung>
<!ATTLIST  #NOTATION     Notationsname Attributdefinition>
```

Die Festlegung des Notationstyps erfolgt in vielen Fällen durch einen Verweis auf das Anwendungsprogramm, das die in der zu definierenden Notation erfassten Daten interpretiert bzw. verarbeitet. Unter dem Gesichtspunkt der softwareunabhängigen Datenhaltung – einem der Hauptziele der Entwicklung von SGML – erscheint eine derartige Notationsdefinition allerdings als nicht sinnvoll.

Notationen können mittels Attributen genauer spezifiziert werden. Die Attributdefinition erfolgt analog zur Attributzuweisung für Elemente, wobei jedoch

zusätzlich das Schlüsselwort `#NOTATION` dem Bezeichner des zu Attributierenden vorangestellt wird.

2.1.3 Dokumentinstanz

Den Ausgangspunkt für eine Verwendung von SGML bildet die Notwendigkeit, Daten in einer Form zu halten, die es erlaubt, aus der Verwendung von Rechnern einen möglichst hohen Mehrwert zu erzielen. Hierzu wird in der SGML-Deklaration das SGML-System konfiguriert und durch die DTD ein Rahmen für die Strukturierung der Daten spezifiziert. Die eigentliche Domäne der Anwendung von SGML bilden somit die zu strukturierenden Daten. Sie stellen die zu repräsentierende Information dar und werden Inhaltsdaten genannt. Alle Inhaltsdaten sind, direkt oder indirekt[4], in der Dokumentinstanz enthalten. Die Dokumenteninstanz enthält neben den Inhaltsdaten auch Metadaten, die u. a. die Inhaltsdaten strukturieren.

2.1.3.1 Inhaltsdaten

Die häufigsten auftretenden Daten sind Textdaten, also Wörter, Zahlen und Sonderzeichen. Sie können im Normalfall von Menschen geschrieben und gelesen werden. Die textuellen Daten stehen im Kontrast zu binären Daten, die oft zur Repräsentation von Bildern, Filmen oder akustischen Signalen verwendet werden. In die Dokumentinstanz sind auch diese binären Daten integrierbar, wobei sich die nichttextuellen Daten jedoch notwendigerweise in eigenen Dateien befinden, die als nicht weiter strukturierte Einheiten *en block* in die SGML-Instanz eingefügt werden. Zwar können die textuellen Daten auch auf mehrere physikalische Einheiten verteilt sein, im Unterschied zu den Binärdaten können sie jedoch weiter strukturiert werden. Die direkte Einbettung der Inhaltsdaten in die Dokumentenstruktur kann auf zwei unterschiedliche Arten erfolgen:

- durch die Einfügung der Inhaltsdaten in diejenigen in der DTD definierten Umgebungen, die – mittelbar oder unmittelbar – als Inhaltsmodell `#PCDATA` akzeptieren oder

[4] Indirekt in der SGML-Instanz enthaltene Daten können dann vorliegen, wenn in der DTD Entitäten definiert wurden. Durch ihre Verwendung in der Dokumentinstanz können die Textteile eingefügt werden. Nach der Verarbeitung mit sogenannten SGML-Normalisierern werden die Primärdaten in die Instanz an der Stelle der Entitätenaufrufe eingefügt.

- als Wert von Attributen, deren Typ in der DTD als `CDATA` definiert wurde.

In beiden Fällen können frei wählbare Abfolgen von 0 bis n Zeichen eingegeben werden. Die frei gewählten Zeichenketten bilden die Abschnitte der Dokumenteninstanz, für die das SGML-System keinerlei Kontrollfunktion ausübt.

Es ist also möglich, vollständig unstrukturierte Texte in SGML-Instanzen zu integrieren, indem sie in ein Element oder ein Attribut eingefügt werden, welches ein unrestringiertes Inhaltsmodell besitzt. (vgl. Witt 1999b) Dies zeigt, dass die Verwendung von SGML per se noch keinen großen Vorteil gegenüber einem simplen Textdatenformat bietet. Erst wenn Metadaten Verwendung finden, werden SGML-Daten zum Gegenstand einer Informationsstrukturierung.

2.1.3.2 Meta-Daten

Als Meta-Daten werden hier alle Informationen bezeichnet, die Inhaltsdaten markieren bzw. strukturieren oder die sie indirekt repräsentieren. Des Weiteren gehören speziellere, hier nur am Rande relevante – und bei Bedarf eingeführte – Einheiten, z. B. die processing instructions, sowie die Konstruktionen, die es erlauben die DTD zu strukturieren, z. B. die soeben vorgestellten Parameterentitäten, zu dieser Gruppe.[5]

Die Meta-Daten werden auch als Markup oder als Annotation bezeichnet. Annotationen sind leicht zu erkennen: sie beginnen immer mit bestimmten Zeichen(-ketten), die in der SGML-Deklaration definiert werden. Im Standardfall, d. h. in standardisierten SGML-Deklarationen (von deren Syntaxkonventionen hier ausgegangen wird) sind dies die Zeichen `<`, `&` und `%`. Die Markierung und die indirekte Repräsentation von Inhaltsdaten bilden die zentralen Verwendungsweisen der Meta-Daten.

> **Markierungen von Inhaltsdaten** erlauben eine direkte Repräsentation dieser Daten. Sie resultieren aus der Verwendung von SGML-Elementen, deren in der DTD festgelegtes Inhaltsmodell das Schlüsselwort `#PCDATA` enthält oder aus der Vergabe eines Attributnamens

[5] Der Begriff Meta-Daten wird auch für Informationen über Informationen gebraucht, beispielsweise das Datum der Erstellung eine Dokumentes. Diese andere Art von einem Meta-Datum kann durchaus rein textuell, d. h. als Primärdatum im obigen Sinne, vorliegen.

für einen textuellen Datenwert. Ein Beispiel für ein derartiges Attribut bildet eines, dem in der DTD der Typ `CDATA` zugeordnet wurde und welches unrestringierte Daten aufzunehmen erlaubt. Es gibt auch Attribute zur Repräsentation von Inhaltsdaten, die den Typ des Wertes restringieren. Zu dieser Klasse zählt z. B das Schlüsselwort `NUMBER`, welches besagt, dass der Attributwert eine Zahl sein muss.

Indirekte Repräsentationen von Inhaltsdaten liegen dann vor, wenn in der DTD bereits eine Einschränkung möglicher Werte vorgenommen wurde. Dies geschieht meist auf zwei unterschiedliche Arten: durch leere Elemente oder durch die Vergabe eines Attributwertes, der aus einer in der DTD festgelegten Auswahlliste stammt.

Eine Aufgabe des DTD-Designs besteht darin, zu entscheiden, welche Daten in einer direkten und welche in einer indirekten Weise erfasst werden sollen. Direkt repräsentierte Daten besitzen den Vorteil, besser von Menschen erfasst und verarbeitet werden zu können, indirekt repräsentierte Daten sind für Maschinen geeigneter.

Das gesamte in der DTD definierte Repertoire von Elementen, Attributen und Entitäten bildet die Bestandteile der – in der Dokumentinstanz zur Verfügung stehenden – Metadaten. Elemente werden durch den in spitze Klammern eingeschlossenen Elementnamen (`<element>`) zur Markierung eines Umgebungsbeginns und durch spitze Klammern inklusive eines Schrägstrichs (`</element>`) zur Markierung eines Umgebungsendes in der Dokumentinstanz markiert bzw. ausgezeichnet. Für Elemente mit einem als `EMPTY` definierten Inhaltsmodell wird nur der Umgebungsbeginn markiert, das Umgebungsende darf nicht annotiert werden. Die Attribute werden als Attribut-Wert-Paar (`attribut='wert'`) in die Markierung des Umgebungsbeginns, zwischen dem Elementnamen und der schließenden Klammer, eingefügt. Die Reihenfolge der Attribute ist irrelevant. Entitäten werden – analog zur Markierung von Parameterentitäten – in ein Kaufmannsund (`&`) und ein Semikolon (`;`) eingeschlossen.

2.1.3.3 Struktur der Dokumentinstanz

Dokumenteninstanzen können unterschiedlich komplex sein, somit auch ihre Struktur. Die einfachste denkbare SGML-Instanz besteht aus genau einem Element mit einem leeren Inhaltsmodell ohne Attribute. Die Dokumentinstanz

besteht aus einem Verweis auf die DTD (vgl. Abschnitt 2.1.4) und der Markierung des Elementbeginns:

```
(Verweis auf DTD)
<a>
```

Dieses Dokument kann als Baum, bestehend aus genau einem Knoten, der Wurzel, repräsentiert werden.

Abbildung 1: Wurzelelement

Jedes SGML-Dokument muss genau ein Wurzelelement besitzen. Im Normalfall sind innerhalb der durch das Wurzelelement markierten Umgebung Daten enthalten, die in eine Baumrepräsentation als Knoten unter das Wurzelelement eingefügt werden können. Wird durch ein SGML-Dokument eine Zeitungsmeldung repräsentiert, kann die Dokumentinstanz folgendermaßen aussehen:[6]

```
<Zeitungsmeldung><Schlagzeile>Keine Gespräche über
Weimarer Nationaltheater</Schlagzeile>
<Quelle>Berlin (dpa)</Quelle>
<Meldung><Abschnitt>Der Bund will nach Informationen von
... </Abschnitt></Meldung></Zeitungsmeldung>
```

Eine Baumrepräsentation dieser SGML-Instanz ist in der Abbildung zu sehen:

[6] Auf die Angabe des Verweises auf die Dokumentenstruktur wird nachfolgend verzichtet.

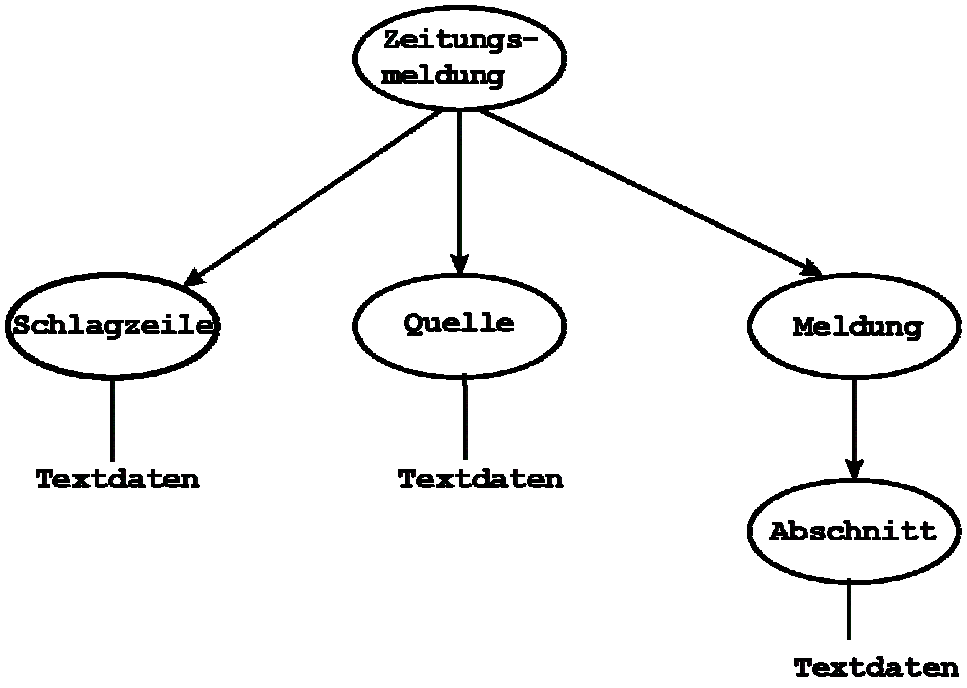


Abbildung 2: Baumstruktur einer SGML-Instanz

Der Baum spiegelt die Struktur des Ausgangsdokumentes wider, einer Schlagzeile folgt eine Quellenangabe, der die weiter strukturierte Meldung folgt. Es ist erkennbar, dass die Reihenfolge der Knoten eines eine SGML-Instanz repräsentierenden Baumes relevant ist. Derartige Bäume sind mit den in der Linguistik verwendeten Bäumen zur Repräsentation der grammatischen Struktur von Sätzen vergleichbar (vgl. u. a. Chomsky 1957).

Es ergibt sich aus den in einer DTD definierbaren Inhaltsmodellen, dass die in der Baumrepräsentation knotenbildenden Elemente notwendigerweise entweder aufeinanderfolgen oder vollständig ineinander verschachtelt sind – Umgebungen können sich nicht überlappen. (s. Kapitel 1)

Die Repräsentation muss es allerdings auch erlauben, den Knoten die Attribute hinzuzufügen, wobei beachtet werden muss, dass sich der ontologische Status der Attribute von denen der Tochterelemente unterscheidet, so dass die Baumknoten nicht einfach um weitere – den direkten Kindknoten nebengestellte – Knoten ergänzt werden können. Die beste Möglichkeit den Elementen Attribute zuzuordnen, besteht darin, in die Elementknoten die Repräsentation der Attributliste einzufügen. Ein Typ der Attribute – `IDREF(S)` (Abschnitt 2.1.2.2) – führt jedoch dazu, dass sich eine derartige Baumrepräsen-

tation formal nicht aufrechterhalten lässt, da von jedem Knoten auf beliebig viele andere Knoten verwiesen werden kann. Dies kann durch Bäume nicht ausgedrückt werden. Auf diese Detailliertheit soll allerdings hier verzichtet werden, da, wenn immer auf derartige Referenzen verzichtet wird, die Baumrepräsentation ausreicht.

Eine graphische Darstellung der Dokumentbäume ist jedoch für die meisten aussagekräftigen Dokumentausschnitte so umfangreich, dass in dieser Arbeit nach Möglichkeit auf derartige Re-Repräsentationen verzichtet wird. Wenn notwendig, wird auf eine zu dieser graphischen Repräsentation äquivalenten Klammernnotation zurückgegriffen. Hierfür steht prinzipiell eine Vielzahl von Schreibweisen zur Verfügung, die hier verwendete Darstellung heißt ESIS und zählt zu den ältesten normierten SGML-Repräsentationen.[7]

2.1.4 Zusammenspiel der Komponenten – logische und physikalische Verteilung der Ressourcen

Die unterschiedlichen Teile eines SGML-Dokuments werden miteinander verknüpft. Die Assoziation der SGML-Deklaration mit den anderen Komponenten erfolgt implizit bzw. softwareabhängig, was in der Praxis bedeutet, dass die Deklaration entweder den anderen Komponenten des Dokumentes - möglicherweise in derselben Datei - vorangestellt wird oder in einer separaten Datei gehalten wird, deren Lokation dem Anwendungsprogramm mitgeteilt wird. Wenn die SGML-Deklaration fehlt, dann wird, wie bereits erwähnt, auf die RCS zurückgegriffen. Es ist also u. U. möglich, alle Komponenten des SGML-Dokumentes in einer einzigen physikalischen Datei zu speichern. Ein Umstand, der ein solches Vorgehen nicht erlaubt, ist die Verwendung von Daten, die nicht in SGML repräsentiert werden, die z. B. bei der Integration von Audio- oder Videosequenzen benötigt werden.

2.1.4.1 DTD - Dokument

Die Verbindung der Dokumenteninstanz mit der Grammatik kann, abhängig von der verwendeten Software, ebenfalls auf unterschiedliche Art und Weise

[7] "The set of information that is acted upon by implementations of structure-controlled applications is called the 'element structure information set' (ESIS). ESIS is implicit in ISO 8879, but is not defined there explicitly." (Goldfarb 1990: 588). Eine explizite Definition bzw. Ausführung der impliziten im SGML-Standard vorhandenen Information über diese Informationseinheit gibt Goldfarb selbst im Anschluss an die zitierte Erläuterung.

erfolgen, jedoch sind Standardvorgehensweisen definiert. Eine SGML-Instanz beginnt mit der Bestimmung des Dokumententyps:

```
<!DOCTYPE Dokumentenname (DTD-Verweis ?[Explizite DTD])>
```

Der Name des Dokumentes muss genauso lauten wie der Name des ersten Elementes. Im Anschluss erfolgt die Angabe der DTD.

2.1.4.2 Explizite DTD

Nach der Angabe des Dokumentennamens erfolgt in eckigen Klammern (zwischen `[´ und `]´) die Angabe der DTD. Als Beispiel soll ein sehr wenig strukturiertes Dokument dienen, welches die Dokumentengrammatik direkt integriert, also explizit angibt:

```
<!DOCTYPE text [<!ELEMENT text  (#PCDATA)
                ]>
<text>Ich mag keine strukturierten Texte.</text>
```

2.1.4.3 Verweis auf eine externe DTD

Ihrem Wesen nach dient eine Document Type Definition der Beschreibung einer Klasse von gleich oder ähnlich strukturierten Texten. Daraus folgt, dass meist eine DTD für mehrere Instanzen verwendet wird. Es ist daher sinnvoller, nicht bei jedem Gebrauch dieser Grammatik eine Kopie der DTD in den mit `<!DOCTYPE` beginnenden Abschnitt des Dokumentes zu integrieren, sondern diese DTD separat zu halten und von den Instanzen auf sie zu verweisen. Dies kann auf zwei verschiedene Arten erfolgen - mittels rechnerspezifischen und öffentlichen Verweisen. Die technischen Termini sind die aus dem Englischen übernommenen Begriffe system identifier(s) respektive public identifier(s). Es folgt ein Beispiel des Verweises mit einem system identifier:

```
<!DOCTYPE text  SYSTEM „pfad/datei.dtd“> (UNIX) oder
<!DOCTYPE text  SYSTEM „Laufwerk:\pfad\datei.dtd“>
(winDOS)
```

Die Sprache XML (s. u.) beendet diese rechner- bzw. betriebssystemspezifische Varianz, da auf Systemdateien in einer genormten Weise – mittels URIs – verwiesen wird.

Ein öffentlicher Verweis hat eine festgelegte Form (Open Technical Resolution TR9401:1995). Diese wird nach Schlüsselwort PUBLIC an der Stelle ein-

gesetzt, an der sich im Fall der system identifiers der Verweis auf die Lokation der Ressource befindet. Als Beispiel soll der Verweis auf eine Fassung der Sprache HTML dienen:

```
<!DOCTYPE HTML PUBLIC „-//W3C//DTD HTML 3.2//EN“>
```

Öffentliche Identifikatoren werden insbesondere dazu verwendet, um auf standardisierte DTDs zurückzugreifen.

2.1.4.4 Mischform

Wie in so vielen Fällen, in denen zwei Möglichkeiten zur Auswahl stehen, gibt es auch hier eine dritte, nämlich die Mischung beider. Dies bedeutet konkret, dass sowohl ein Verweis auf eine externe DTD, als auch ein interner DTD Anteil angegeben wird. Besondere Bedeutung hat dies für die Verwendung von Entitäten, die mehrfach definiert werden dürfen. In einem solchen Fall erhält die mehrfach definierte Entität den Wert der ersten Belegung. Da ein SGML-Programm als erstes den direkt angegebenen Teil der DTD lesen muss, können Entitäten verändert werden, ohne die externe, möglicherweise standardisierte DTD zu verändern.

2.2 Probleme in SGML

2.2.1 Ambiguität von Inhaltsmodellen

Durch eine Dokumenttypdefinition werden Restriktionen der verschiedenen möglichen Dokumentinstanzen aufgebaut. Eine DTD ist eine formale Grammatik, die u. a. restringiert, welche Abfolgen von Elementen in welchen Umgebungen verwendet werden dürfen. So kann in einer DTD z. B. sehr einfach definiert werden, dass ein Absatz aus mindestens drei Sätzen bestehen muss:

```
<!ELEMENT Abschnitt (Satz, Satz, Satz+)>
<!ELEMENT Satz      (#PCADTA)>
```

Wenn jedoch ausgedrückt werden soll, dass es auch eine maximale Anzahl von aufeinanderfolgenden Sätzen geben soll, kann dies nicht in derselben Weise erfolgen, da SGML die sogenannten ambigen Inhaltsmodelle verbietet. Die naheliegendste Möglichkeit einer derartigen Restriktion, z. B. dass ein

Paragraph aus mindestens aus mindestens 2 und maximal aus 7 Sätzen bestehen darf, führt zu einem solchen ambigen Inhaltsmodell:

```
<!ELEMENT Abschnitt (Satz, Satz, Satz?, Satz? , Satz?,
                     Satz?, Satz?)>
<!ELEMENT Satz      (#PCADTA)>
```

Wird nun z. B. ein aus sechs Sätzen bestehender Abschnitt geparst, kann nicht zugeordnet werden, ob der im obigen Inhaltsmodell des Elementes Abschnitt an dritter, vierter oder sonstiger Stelle auftretende `Satz` weggelassen wurde.

Eine Möglichkeit dies Ambiguität aufzulösen besteht darin, mit Gruppierungen zu arbeiten, d. h.:

```
<!ELEMENT Abschnitt (Satz, Satz, (Satz, (Satz, (Satz ,
                                  (Satz, Satz?)?)?)?)?)>
<!ELEMENT Satz      (#PCADTA)>
```

Es ist zu erkennen, dass eine derartige ambiguitätsfreie Schreibweise allerdings – insbesondere bei umfangreicheren Inhaltsmodellen – die Lesbarkeit der DTD in einer gravierenden Weise erschwert.

Zu diesem Problembereich gibt es eine umfangreiche Liste von Arbeiten, wobei an erster Stelle die Arbeiten von Anne Brüggemann-Klein genannt werden müssen (z. B. Brüggemann-Klein 1993), die u. a. zeigt dass Überführungen von ambigen Inhaltsmodelldefinitionen in äquivalente ambiguitätsfreie Versionen nicht immer möglich sind.

2.2.2 Inferenz von Elementbezeichnern

SGML erlaubt, wie oben bereits beschrieben (vgl. S. 15), das Auslassen der Markierung des Beginns und/oder des Endes von Umgebungen. Wird in der SGML-Deklaration dem Schlüsselwort `OMITTAG` der Wert `YES` zugewiesen, verändert sich die Elementdefinition (vgl. S. 17) dahingehend, dass zwischen dem Elementnamen und dem Inhaltsmodell angegeben werden muss, ob die Markierung weggelassen werden darf oder nicht.

In HTML-Dokumenten muss z. B. das Ende eines Abschnitts nicht markiert werden, da die HTML-DTD das Element `<p>` wie folgt definiert:

```
<!ELEMENT p – o Inhaltsmodell>
```

Dieses Mittel diente ursprünglich der Reduktion des Schreibaufwands bei der Erstellung von SGML-Instanzen.[8]

Die nachfolgenden Ausführungen sollen jedoch zeigen, dass die Inferenz von Elementen bzw. Umgebungen einen Wert darstellt, der auch aus einer computerlinguistisch motivierten Dokumentverarbeitungsperspektive äußerst sinnvoll sein kann. Bedauerlicherweise wird sich herausstellen, dass die prinzipielle Möglichkeit der Inferenz von Elementbezeichnern so stark eingeschränkt wurde, dass sie nicht gewinnbringend genutzt werden kann.

Eine vorstellbare sinnvolle Nutzung dieses Merkmals besteht in der Inferenz von Phrasen. Das folgende SGML-Dokument soll hierfür als Beispiel dienen. Die DTD definiert die Phrasenmarkierungen als auslassbar:

```
<!ELEMENT text  o o   (s)+>
<!ELEMENT s     o o   (np, vp)>
<!ELEMENT np    o o   (det, n)>
<!ELEMENT vp    o o   (v, np)>
<!ELEMENT (n|v|det|en)  - -  (#PCDATA)>
```

Eine vollständige mögliche Dokumentinstanz ist:

```
<det>der</det><n>Mann</n><v>schnitzte</v><det>eine</det><n>Keule</n>
```

Eine SGML-Verarbeitung inferiert die ausgelassenen Markierungen und repräsentiert sie intern vollständig als:[9]

```
<text><s><np><det>Der</det><n>Mann</n></np>
         <vp><v>schnitzte</v><np><det>eine</det>
                                 <n>Keule</n></np></vp></s></text>
```

Der SGML-Standard erlaubt allerdings nur die Inferenz der Elementbezeichner, die als „contextually required“ bezeichnet werden, also Elemente die unter allen Umständen an dieser Stelle vorkommen müssen, ein Spezialfall. Wäre z. B. in der DTD die Regel der Verbalphrase realistischerweise so for-

---

[8] Insbesondere bei sehr einfachen Editoren, die zur Zeit der Entwicklung von SGML das wichtigste Dateneingabewerkzeug waren, kann es praktisch sein, weniger MarkUp tippen zu müssen. Seit geraumer Zeit werden für diese Zwecke jedoch sehr komfortable spezielle Editoren verwendet, wodurch diese Anwendung in den Hintergrund tritt.

[9] Diese vollständige bzw. normalisierte Dokumentinstanz kann durch die Verwendung sogenannter SGML-Normalizer maschinell aufgebaut werden.

muliert worden, dass auch intransitive Verben verwendet werden könnten[10], wäre das Element `np` nicht „contextually required“, wodurch die obige Inferenz nicht möglich wäre.

Eine andere wünschenswerte Inferenz ist auch dann nicht möglich, wenn das Element „contextually required“ ist: die Inferenz leerer Elemente. Wenn der obigen Satzdefinition ein Element zur Markierung des Interpunktionszeichens für das Satzende hinzugefügt werden soll, kann dies mit der leicht veränderten DTD ausgedrückt werden[11]:

```
<!ELEMENT text  o o   (s)+>
<!ELEMENT s     o o   (np, vp, Satzende)>
<!ELEMENT np    o o   (det, n)>
<!ELEMENT vp    o o   (v, np)>
<!ELEMENT Satzende o o (#PCDATA)>
<!ELEMENT (n|v|det|en)  - -  (#PCDATA)>
```

Bei der Verarbeitung des Textes:

```
<det>der</det><n>Mann</n><v>schnitzte</v><det>eine</det><n>Keule</n>.
```

wird dann die Annotation `<s>...</n></vp><Satzende>.<Satzende>` aufgebaut. Da das in diesem Fall verwendbare Interpunktionszeichen ausschließlich ein Punkt sein kann, wäre es aber für diesen Zweck sinnvoller ein leeres Element mit einem Festwert zu verwenden (vgl. S. 19), dieses wegzulassen und im Zuge der Verarbeitung zu inferieren. Diese sinnvolle Verwendungsweise der Elementinferenz, die z. B. auch zur automatischen Ermittlung und Einfügung von Leerzeichen zwischen annotierten Wörtern verwendet werden könnte, ist vom SGML-Standard explizit untersagt.[12]

---

[10] Ein derartige Elementdefinition kann dann die folgende Form besitzen: `<!ELEMENT vp o o (v, np?)>`

[11] Die Definition `<!ELEMENT (n|v|det|en) - - (#PCDATA)>` ist eine in SGML erlaubte Zusammenfassung von Elementen mit demselben Inhaltsmodell.

[12] Der SGML-Standard besagt “The *start-tag* can be omitted ...except if ... the *content* of the instance of the element is empty.” (ISO 8859:22-23)

2.2.3 Mixed-Content

Nichtleere Inhaltsmodelle von Elementen können weitere Elemente oder Textdaten (`#PCDATA`) enthalten – oder eine Mischung aus beidem, dem mixed-content[13].

Die wichtigste Problematik bei der Verwendung gemischter Inhaltsmodelle besteht darin, dass in diesem Fall die Trennzeichen, die eingefügt werden, wenn Elemente zum Zwecke der besseren Lesbarkeit voneinander abgrenzt werden, als `#PCDATA` interpretiert werden, selbst wenn z. B. einer Sequenzen von Elementen unrestringierte Daten nachfolgen dürfen. Wird z. B. ein Inhaltsmodell aufgebaut, das die Abfolge von ein oder zwei Sätzen legitimiert und das besagt, dass diesem Inhalt nicht in Umgebungen eingeschlossener Dateninhalt folgen darf, kann dies durch die folgende DTD erfolgen:[14]

```
<!ELEMENT minitext    (s, s?, #PCDATA)>
<!ELEMENT s           (#PCDATA)>
```

Allerdings wird durch diese DTD die nachfolgende Dokumenteninstanz, entgegen der naheliegenden Interpretation, nicht legitimiert.

```
<minitext>
   <s>Der Satz 1 ...</s>
   <s>Der Satz 2 ...</s> Was noch zu sagen ist
..</minitext>
```

Der Grund hierfür ist im SGML-Standard zu suchen, in dem angegeben wird, dass in Inhaltsmodelle, in denen `#PCDATA` vorkommen darf, Zeichen, die auch als Elementseparatoren verwendet werden können (z. B. Leerzeichen oder Zeilenwechsel), hier als Daten interpretiert werden müssen. D. h. im obigen Beispiel, dass – entgegen jeglicher Intuition – zwischen der Markierung `<minitext>` und `<s>` Daten (der Zeilenwechsel und die Leerzeichen) erkannt werden, die gemäß des Inhaltsmodells an dieser Stelle nicht auftreten dürfen.

---

[13] Der Begriff mixed-content bezeichnet korrekterweise, d. h. im ISO-Standard, auch Inhaltsmodelle, die nicht „gemischt“ sind, sondern ausschließlich aus `#PCDATA` bestehen. In dieser Arbeit bezeichnet der Terminus jedoch ausschließlich Mischungen von `#PCDATA` und Elementen.

[14] Es wird, im Gegensatz zum vorangegangenen Abschnitt, wieder angenommen, dass `OMITTAG` den Wert `NO` zugewiesen bekommt.

Um innerhalb von gemischten Inhaltsmodellen[15] die Annotationen so aufzubereiten, dass sie auch Menschen übersichtlich und relativ leicht lesbar erscheinen, kann eine beliebige Anzahl von Trennzeichen unmittelbar bevor die Elementmarkierung beendet wird oder zwischen Attributen eingefügt werden:

```
<minitext
><s>Der Satz 1 ...</s
><s>Der Satz 2 ...</s
> Was noch zu sagen ist ..</minitext>
```

Diese korrekte, wenn auch selten verwendete, Annotationsform wird im Verlauf dieser Arbeit bei Bedarf gewählt.

2.2.4 Verwendung einer DTD

Wie schon gezeigt ist ein SGML Dokument dann unvollständig, wenn ihm keine DTD beigefügt wurde. Derartige Dokumente können jedoch u. U. eine sehr gut verwendbare Datengrundlage sein – zumindest bilden sie zweifelsohne eine wesentlich bessere Basis für eine maschinelle Verarbeitung als unannotierte Textdokumente. Die Ursachen hierfür bestehen insbesondere darin, dass die Strukturierungsinformationen zwar einerseits in den Text eingearbeitet sind, andererseits jedoch durch die Verwendung der speziellen Notation klar von ihm abgegrenzt sind. Des Weiteren stellt die Einfügung der Strukturierung an sich einen Vorteil dar, z. B. weil die Speicherung in Datenbanken verbessert wird. Es ist darüber hinaus sehr häufig wünschenswert, Daten, d. h. Dokumentinstanzen auszutauschen, obwohl das in die DTD eingefügte Wissen nicht weitergegeben werden soll.

Ein weiterer gegen einen Austausch der DTD sprechender Punkt besteht darin, dass in vielen Fällen schlicht überflüssig ist, auf sie zurückzugreifen. SGML-Dokumente sollen gedruckt oder an Bildschirmen angezeigt werden. Dafür werden notwendigerweise verschiedene weitere Komponenten, insbesondere sogenannte Style-Sheets benötigt (vgl. z. B. Sasaki und Witt 2001), jedoch keine DTDs.

SGML weist jedoch SGML-Instanzen, denen eine DTD fehlt, keinen speziellen Status zu, sondern behandelt sie schlicht wie nicht SGML-konforme Dokumente.

---

[15] Das nachfolgend beschriebene Vorgehen ist nicht auf die soeben erwähnten ‚unintuitiven' mixed-content models beschränkt.

2.2.5 Komplexität

Durch die drei ein SGML-Dokument konstituierenden Komponenten und der darin möglichen vielfältigen Anpassungsmöglichkeiten ist eine hohe Komplexitätsstufe erreicht worden. Wenngleich durch all die Adaptionsoptionen ein sehr flexibles und mächtiges Instrumentarium zur Verfügung steht, führt die hierfür notwendige Komplexität zu zwei negativen Resultaten: erstens können u. U. korrekte SGML-Dokumente für Menschen sehr schwer lesbar sein und zweitens ist es außerordentlich aufwändig SGML-verarbeitende Software zu erstellen, die nicht nur eine Teilmenge aller möglicher SGML-Dokumente verarbeiten können soll.

Insbesondere die Komplexität, aber auch die weiteren, oben genannten Punkte führten dazu, dass SGML überdacht wurde und dass mehr als zehn Jahre nach Verabschiedung des ISO-Standards von einem anderen Gremium – dem World Wide Web Consortium (W3C), ein einfacherer, stark an SGML-orientierter Standard, die eXtensible Markup Language (XML, Bray et al. 2000[16]) verabschiedet wurde.

2.3 Die Syntax von XML

Die extensible Markup Language vereinfacht SGML in vielerlei Hinsicht. Im Prinzip sind alle korrekten XML-Dokumente auch korrekte SGML-Dokumente, wobei die Einschränkung darin besteht, dass der SGML-Standard leicht modifiziert werden musste, um sicherzustellen, dass dieser Anspruch auch eingelöst werden konnte. Nach dieser Modifikation besteht nicht mehr nur prinzipiell eine Untermengenbeziehung zwischen XML und SGML, d. h. es gilt: jedes XML-Dokument ist auch ein SGML-Dokument, nicht jedoch umgekehrt.

Der XML-Standard trifft eine fundamentale Unterscheidung der XML-Dokumente, es werden wohlgeformte und gültige XML-Dokumente definiert. Mit der Einführung des in SGML unbekannten Konzeptes der Wohlgeformtheit wird erreicht, dass annotierte Dokumente, denen eine DTD fehlt, einen eigenen formalen Status erhalten und dadurch separat verarbeitbar und austauschbar werden.

Ein wohlgeformtes XML-Dokument enthält Inhaltsdaten und Annotationen. Die Annotationen zeichnen Umgebungen aus, die mit Attributen versehen sein können. Wie bei der oben beschriebenen Standardverwendung von

[16] Die zitierte Version stellt eine Überarbeitung der ersten als Standard verabschiedeten Version dar, die aus dem Jahr 1998 stammt.

SGML beginnt eine Umgebung auch in XML mit <...> und endet mit </...>. Dies gilt jedoch im Gegensatz zum ‚alten' SGML für alle Elemente, auch die ohne Inhalt. Allerdings darf eine solche Umgebung statt <n></n> auch verkürzt <n/> geschrieben werden.

Ein wohlgeformtes XML-Dokument muss genau eine alles umschließende Umgebung besitzen – in der Baumrepräsentation der Wurzelknoten – und die Umgebungsgrenzen dürfen sich nicht überschneiden.

Ein gültiges XML-Dokument besteht nicht nur aus der Dokumentinstanz sondern besitzt auch eine DTD. Ein XML-Dokument wird dann als gültig bezeichnet, wenn es nicht gegen die den Dokumenttyp definierende Grammatik verstößt. Die in der DTD verwendbaren Konstrukte stellen eine Untermenge der in SGML-DTDs zur Verfügung stehenden Möglichkeiten dar. Die wichtigsten Einschränkungen der XML-DTDs bestehen in den folgenden Punkten:

- Das Zeichen & (vgl. S. 18) steht zur Definition von Inhaltsmodellen nicht zur Verfügung.
- Es ist nicht möglich, Elementnamen als inferierbar zu definieren (vgl. Abschnitt 2.2.2).
- Gemischte Inhaltsmodelle dürfen nur in der Form (PCDATA|...)* definiert sein. Dies vermeidet die im Abschnitt 2.2.3 beschriebenen Probleme.
- Alle Elemente müssen einzeln definiert sein. Zusammenfassungen von Elementen mit dem selben Inhaltsmodell sind nicht zulässig (vgl. Fußnote 11 S. 32).
- Attributlisten dürfen nicht für Notationen verwendet werden (vgl. Abschnitt 2.1.2.4 ab S. 21).

Der XML-Standard erlaubt es nicht, von der vorgegebenen Syntax abzuweichen. Während die SGML-Dokumentinstanzen und die SGML-DTD ein (vereinfachtes) Analogon in XML besitzen, existiert in XML kein mit der SGML-Deklaration vergleichbares Konstrukt.

All diese Vereinfachungen führen dazu, dass XML-Dokumente wesentlich weniger komplex sind als SGML-Dokumente. Dies ist vermutlich der wichtigste Umstand, der dazu beitrug, dass XML innerhalb kürzester Zeit außerordentlich populär wurde und festgestellt werden muss, dass SGML bedeutungslos geworden ist.

In der hier vorliegenden Arbeit wird versucht, ausschließlich XML-konforme SGML-Konstrukte zu verwenden, nichtsdestotrotz wird häufig auf den wesentlich älteren Standard SGML Bezug genommen werden. Der Grund hierfür ist einerseits, dass sich die Vorarbeiten z. T. auf SGML beziehen, da XML noch nicht angedacht war, andererseits wäre es äußerst ignorant die bereits vor geraumer Zeit standardisierten Möglichkeiten nicht zu diskutieren, wenn Limitierungen der XML-basierten Auszeichnungsmodelle thematisiert werden.

2.4 Schema-Sprachen

SGML- oder XML-Dokumente werden durch Dokumentstrukturbeschreibungen, die Dokumentgrammatiken, restringiert. Sowohl in SGML als auch in XML ist mit der Document Type Definition für diese Zwecke ein Format festgelegt, das es erlaubt, diese Restriktionen auszudrücken. Sprachen, die es erlauben Dokumentgrammatiken zu spezifizieren, werden in jüngerer Zeit auch (Dokument-)Schema-Sprachen genannt. Alle SGML-Dokumente und alle gültigen XML-Dokumente müssen eine in der Schema-Sprache DTD ausgedrückte Dokumentgrammatik besitzen.

Durch die Einführung des Konzeptes der Wohlgeformtheit ist es jedoch auch möglich, XML-Dokumente ohne DTDs zu verarbeiten. Es war im Abschnitt 2.2.4 zu sehen, dass dies diverse Vorteile mit sich bringt. Allerdings scheint die Einführung dieses Konzeptes aus der Perspektive der Informationsmodellierung einen Rückschritt darzustellen, da Dokumentbeschreibungen die Möglichkeit der abstrakten Informationsstrukturierung und der Bezugnahme auf ganze Dokumentklassen bieten. Auf der anderen Seite erlaubt es allerdings dieses Konzept auch – statt auf eine formale Dokumentmodellierung zu verzichten – eine verbesserte Dokumentmodellierung einzuführen. Dies kann dadurch erreicht werden, dass nicht nur auf die Angabe der nicht mehr obligatorischen DTD verzichtet wird, sondern andere Methoden der Dokumentrestriktion zur Anwendung gebracht werden.

In neuerer Zeit sind für die Zwecke der Dokumentmodellierung verschiedene Schema-Sprachen vorgeschlagen worden, die die DTDs ersetzen können. Gemeinsam ist allen neueren Schemasprachen, dass sie zur Definition der Dokumentgrammatik der XML-Instanzen ebenfalls XML verwenden, d. h. eine andere Syntax als die älteren DTDs. Grundsätzlich lässt sich sagen, dass sich die meisten der Restriktionen der Dokumentstruktur, die mittels einer DTD ausgedrückt werden können, ebenfalls mittels der anderen derzeit diskutierten Schemasprachen ausdrücken lassen – nicht jedoch umgekehrt.

Die durch DTDs modellierbaren Dokumentstrukturen sind im Wesentlichen durch kontextfreie Grammatiken beschreibbar. Die Ausdrucksmächtigkeit der anderen grammatikbasierten[17] Schemasprachen, z. B. XML-Schema (Thompson et al. 2001) und vor allem RELAX NG (Clark und Murata 2001a) ist z. T. wesentlich größer (vgl. Murata et al. 2001).

Die Schemasprache RELAX NG widmet sich im Gegensatz zu der Schemasprache des W3C auch dem oben beschriebenen Problem der Ambiguität von SGML (vgl. Absatz 2.2.1), ambige Inhaltsmodeldefinitionen werden schlichtweg durch diese Schemasprache nicht ausgeschlossen, d. h. sie sind zugelassen.[18]

2.5 Namensräume

Wie schon an der Erwähnung einiger Arbeiten zu erkennen war, verabschiedete die (u. a.) XML-standardisierende Organisation W3C im Gefolge von XML eine Vielzahl weiterer Normen. Eine dieser Normierungen trägt den Namen XML-Namespaces (Bray et al. 1999) und definiert sogenannte Namensräume.

Das Konzept der Namensräume ist jedoch in den XML-Standard nur partiell integriert worden. Prinzipiell ist es zwar möglich, Namensräume in XML-Dokumente zu integrieren, jedoch ist diese Integration rein syntaktischer Natur: klassische XML-Dokumente können von dem Konzept der Namespaces nicht profitieren. Dies liegt insbesondere an der Verwendung von DTDs als Dokumentgrammatik, die – wie zu sehen war – weitaus älteren Datums sind als die Namensräume. In die neueren Schemasprachen sind Namensräume hingegen integriert.

Der den Namensräumen zugrundeliegende Anspruch bestand darin, dass Dokumentgrammatiken in einem möglichst starken Ausmaß wiederverwendbar sein sollen. Zwar liegt es in der Natur der DTDs, dass sie nicht nur für einzelne spezielle Instanzen erstellt sind, sondern für eine Klasse strukturähnlicher Dokumente. Allerdings ist es nicht möglich ausschließlich auf Teile

[17] Es existieren auch nicht grammatikbasierte Schemasprachen wie Schematron (Jelliffe 2001) und Examplotron (van der Vlist 2001)

[18] Die Überwindung dieses Problems hat allerdings seinen Preis. Dieser – wenn auch in der Praxis selten unumgängliche – Fall ambiger Inhaltsmodelle erschwert die Herstellung der Kompatibilität zwischen DTDs und RELAX NG Dokumentgrammatiken. (vgl. Clark und Murata 2001b)

einer DTD zuzugreifen und diese Strukturierungen bei Bedarf in Strukturierungen einer anderen Dokumentgrammatik zu integrieren.[19]

Durch weitverbreitete Verwendung von HTML ergab es sich, dass die Bedeutung vieler der dort definierten Elemente oder auch Attribute allgemein bekannt wurde. Bei der Entwicklung anderer DTDs war es wünschenswert diese Elemente zu verwenden, wenn die entsprechenden – durch sie definierten Bedeutungen – Verwendung finden sollten. So spräche z. B. nichts dagegen, auch in annotierten Dokumenten, die nicht den Dokumenttyp HTML besitzen, für Aufzählungen auf die HTML-Elemente `<ol>` und `<ul>` zurückzugreifen, ohne sie nochmals definieren zu müssen. Die Einführung der Namensräume erlaubt genau dies.

Die Funktionsweise von Namensräumen ist äußerst einfach: standardisierten Dokumentgrammatiken wird ein Namespace zugeordnet, auf den dann von beliebigen Dokumentinstanzen Bezug genommen werden kann. Dies geschieht indem für diesen Zweck ein in der Dokumentinstanz eindeutiges Präfix definiert wird. Dieses wird dann durch einen Doppelpunkt (`:`) abgeteilt den entsprechenden Elementen und Attributen zugeordnet.

Namespaces können zu einem immens wichtigen Mittel der Informationsmodellierung werden, da sie es erlauben auf partielle Dokumentgrammatikdefinitionen zuzugreifen und es dadurch ermöglichen sinnvolle Teile verschiedener Dokumentgrammatiken herauszugreifen und zu neuen Dokumentgrammatiken zu kombinieren.

Ein Nachteil der Namensräumen ist jedoch, dass sie aufgrund ihres Entwicklungszeitpunktes kein integraler Bestandteil von XML (oder gar von SGML) sind. Dies liegt insbesondere daran, dass sie nicht in die Schemasprache DTD integriert sind. Nahezu alle anderen Schemasprachen erlauben die Nutzung des Konzeptes der Namensräume.

Namensräume erlauben es, ein Dokument entsprechend verschiedener Dokumentgrammatiken zu strukturieren. Allerdings sind diesen multiplen Strukturierungen Grenzen gesetzt: Ein XML Dokument kann – auch durch die Verwendung von Namensräumen – keine uneingeschränkten multiplen Struktu-

[19] Dies ist zumindest dann nicht möglich, wenn die DTDs nicht speziell für diese Verwendungsweise konzipiert wurden. Als Beispiele für derartig konzipierte DTDs können die modularisierte XHTML-DTD (Altheim und McCarron, 2001) oder die TEI-DTD (Sperberg-McQueen und Burnard 1994) gelten, wobei auch in diesen DTDs nur in beschränktem Ausmaß die Wiederverwendung von Dokumentgrammatikteilen genutzt werden kann.

rierungen leisten, insbesondere weil die Wohlgeformtheit der XML-Instanzen gewährleistet sein muss.

3 Multiple Annotationsebenen

Texte können auf unterschiedlichen Ebenen strukturiert sein. Renear et al. (1996) diskutieren eine der Grundannahmen der Strukturierung von Textdaten: die „OHCO-These“. Diese besagt, dass ein Text eine „ordered hierarchy of content objects“ bildet. Diese angenommene Hierarchie rechtfertigt eine grundlegende Einschränkung von SGML, die sich auch in der schwächsten Form der Dokumentannotation – den wohlgeformten XML-Dokumenten – wiederfindet: die durch Elemente ausgezeichnete Umgebungen dürfen sich nicht überlappen. Im Verlauf der erwähnten Diskussion der OHCO-These wird die These sukzessiv immer weiter abgeschwächt. Mittels Falsifikation durch Gegenbeispiele werden alle – d. h. auch die abgeschwächtesten – Interpretationen der Hierarchie-These verworfen. In Texten können potentiell mehrere unabhängige Hierarchien existieren, deren Begrenzungen sich überlappen. Es kann vermutet werden, dass sich die Hierarchie-These nur entwickeln konnte, da ihr ein sehr enger Textbegriff zugrunde liegt. Texte bestehen aus Kapiteln, Abschnitten, Überschriften etc. Naheliegend war dies insbesondere deshalb, da die Wurzeln der Textannotation im Buchdruck lagen – gedruckte Textteile sind (meist) genau einer derartigen Basishierarchieebene zuzuordnen.

DeRose (1997) bezeichnet den Fall der Notwendigkeit der Annotation von sich überlappenden Umgebungen als einen gelegentlich auftretenden Spezialfall, wobei er allerdings daran beteiligt war, als eine Vielzahl dieser ‚gelegentlichen’ Auftritte zusammengetragen wurde (Durand et al., 1996). Wird nämlich der enge, implizit vorhandene Textbegriff erweitert, entsteht sehr häufig der Bedarf der Möglichkeit der Annotation multipler Hierarchien.

In den Diskussionen der Problematik mehrfacher unterschiedlicher Strukturierungen wird dem Thema der Qualität der zur Annotation verwendeten Einheiten zu wenig Aufmerksamkeit beigemessen. Dies ist ein Grund, weshalb ein Primat der OHCO-These zur Dokumentannotation beobachtet werden kann. Nur wenn eine sehr eingeschränkte Sichtweise auf Textdaten gewählt wird, liegt es nahe, diese maschinell gut zu verarbeitende Datenstruktur zu verwenden.

Auch wenn, wie gesehen, viele die Bedeutung der Repräsentation unterschiedlicherer, potentiell konkurrierender Hierarchien als eher gering einstuften, gab es bisher bereits eine Vielzahl von Lösungsvorschlägen, sei es für

die „occasional times when elements cross each other“ (DeRose 1997:109), sei es aus akademischem Interesse (Sperberg-McQueen und Huitfeldt, 1999).

Die Bedeutung von Lösungen für dieses Problem rückt jedoch immer stärker in den Fokus der Informationsmodellierungsforschung, nicht zuletzt deshalb, da sich durch die Verwendung elektronischer Texte immer häufiger die Notwendigkeit der Annotierung komplexerer strukturierter Textdokumente ergab. Insbesondere um linguistische Informationen in textuelle Daten zu inkorporieren, ist es notwendig, verschiedene Dokumenthierarchien zum Einsatz zu bringen. Die Erkenntnis der Existenz dieser Notwendigkeit manifestiert sich u. a. in dem europäischen Verbundprojekt MATE (Dybkjær 2000). Das Akronym MATE steht für ‚Multilevel Annotation, Tools Engineering’, wodurch deutlich wird, dass in diesem Projekt Annotationen auf mehreren Ebenen situiert sind. Die dort verwendeten Lösungen zur Annotation von Informationen auf verschiedenen Ebenen erlauben die Markierung von nichthierarchischen Umgebungen, also Texten die der OHCO-These nicht genügen.

Das Problem der Modellierung unterschiedlicher Annotationsebenen ist jedoch nicht auf Dokumente beschränkt, die gegen die OHCO-These verstoßen, sondern tritt bereits schon dann auf, wenn bestimmte Umgebungen, aus unterschiedlichen Perspektiven betrachtet, unterschiedliche Elementbezeichnungen erfordern. Das bedeutet, dass die Betrachtung multipler Ebenen der Textauszeichnung auch dann relevant sein kann, wenn aus der Perspektive der syntaktischen Standardmöglichkeiten von SGML keine Notwendigkeit besteht, spezielle Lösungsverfahren einzusetzen.

Verschiedene Optionen, ihre Möglichkeiten, Grenzen und Idiosynkrasien sind der Gegenstand der nachfolgenden Abschnitte. In einer abschließenden Diskussion werden die Vor- und Nachteile der einzelnen Ansätze zusammengebracht und eine in dieser Arbeit favorisierte Lösung diskutiert.

3.1 CONCUR

Der SGML-Standard (ISO:8859) erlaubt die Markierung einer Dokumentinstanz nach verschiedenen Dokumenttypdefinitionen. Dies wird durch die Erlaubnis zur Verwendung einer speziellen Möglichkeit von SGML ermöglicht. In der SGML-Deklaration kann angegeben werden, dass das gleichzeitige (engl.: concurrent) Strukturieren nach unterschiedlichen DTDs erlaubt ist, wenn in der SGML-Deklaration (vgl. Absatz 2.1.1), dem Schlüsselwort `CONCUR` der Wert `YES` zugewiesen wird. In diesem Fall ist es möglich, der

SGML-Instanz mehr als einen Verweis auf Strukturgrammatiken (DTDs) voranzustellen.

```
<!DOCTYPE Silbstruk [
    <!ELEMENT Silbstruk          (Silbe|Blank)*>
    <!ELEMENT Silbe              (#PCDATA)>
    <!ELEMENT Blank              EMPTY>]>

<!DOCTYPE Mstruk  [
    <!ELEMENT Mstruk             (Morphem|Blank)*>
    <!ELEMENT Morphem            (#PCDATA)>
    <!ELEMENT Blank              EMPTY>]>
```

In dem zu annotierenden Dokument wird dann bei der Annotation der einzelnen Elemente auf die entsprechenden Dokumenttypen verwiesen, indem der relevante Dokumenttyp (in Klammern) dem zum Einsatz kommenden Element vorangestellt wird. Das leere Element `Blank`, das in beiden Dokumenttypen eingesetzt wird, wird nicht speziell gekennzeichnet. Im folgenden Beispiel wird die Phrase „auf dem Lande“ entsprechend der morphologischen Segmentierung und der Silbenstruktur annotiert. Die Überschneidung der Hierarchien tritt beim Nomen ‚Lande’ auf.

```
<(Mstruk)Mstruk><(Silbstruk)Silbstruk
 ><(Mstruk)Morphem><(Silbstruk)Silbe
  >auf</(Silbstruk)Silbe></(Mstruk)Morphem><Blank
  ><(Mstruk)Morphem><(Silbstruk)Silbe
  >dem</(Silbstruk)Silbe></(Mstruk)Morphem><Blank
  ><(Mstruk)Morphem><(Silbstruk)Silbe
  >Lan</(Silbstruk)Silbe><(Silbstruk)Silbe
  >d</(Mstruk)Morphem><(Mstruk)Morphem
  >e</(Silbstruk)Silbe></(Mstruk)Morphem
></(Silbstruk)Silbstruk></(Mstruk)Mstruk>
```

In einer lesbareren – nicht SGML-konformen – Darstellung können die unterschiedlichsten Hierarchien durch unterschiedliche Klammerungen dargestellt werden: `[(auf)]_[(dem)]_[(Lan)(d][e]).`

Die Verwendung dieses Konstruktes ist verschiedentlich kritisiert worden. Durand et al. (1996) nennen die folgenden Nachteile der Verwendung dieser optionalen SGML-Möglichkeit:[20]

[20] Durand et al. (1996) schlagen auch eine Alternative zur Verwendung der Möglichkeit CONCUR vor. Unterschiedliche Annotationen sollen hiernach als separate ‚streams' repräsentiert werden und dadurch indirekt den Umgebungen zugeordnet werden. Dieser Ansatz wurde jedoch nicht weiter expliziert oder gar weiterverfolgt.

- Es können keinen partiellen Annotationen vorgenommen werden, da jede der gleichzeitig gültigen Dokumentannotation vollständig sein muss.
- Dateninhalte eines Elementes (tag content), die nur aus der Perspektive einer DTD heraus sinnvoll sind, können nicht gegenüber der anderen Perspektive verborgen werden.
- Es gibt keine Möglichkeit, Wechselwirkungen und Abhängigkeiten zwischen den gleichzeitig verwendeten verschiedenen DTDs auszudrücken.

Ein bisher nicht thematisierter Nachteil, der jedoch mit dem letztgenannten Punkt verbunden ist, besteht in der Beziehung der Annotationen zueinander: durch die Unabhängigkeit der Annotationen zueinander ist die Reihenfolge der zu unterschiedlichen DTDs gehörenden Elemente irrelevant. D. h. (in Anlehnung an das obige Beispiel) zum einen, dass kein Unterschied zwischen der Annotation:

```
<(s1)m><(s2)s>auf</(s2)s></(s1)m>
```

und der Annotation:

```
<(s2)s><(s1)m>auf</(s1)m></(s2)s>
```

besteht. Zum anderen kann nur durch einen maschinellen Interpretationsschritt herausgefunden werden, dass die Morphem- und die Silbengrenze bei dem Wort ‚auf' identisch sind.

Insbesondere jedoch auch aufgrund einiger konzeptueller Nachteile, die sich aus Problemen durch Wechselwirkungen mit einigen anderen spezielleren SGML-Möglichkeiten ergeben, riet der Hauptentwickler des SGML-Standards, Charles Goldfarb, bereits 1990 von einer derartigen Verwendungsweise von `CONCUR` ab:

> 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.
>
> Goldfarb (1990:304)

Auf der anderen Seite kann allerdings festgestellt werden, dass die logischen Strukturierungsmöglichkeiten, die sich durch die Verwendung von CONCUR

ergeben, große Vorteile mit sich bringen können, wie sich im folgenden Zitat zeigt:

> By using concur, the simplicity of interpretational rules found in [...] the basic SGML model [...] can be combined with a powerful language for expressing constraints on document structures (the DTD). The theoretical and practical advantages outweigh the practical disadvantages and the humanities computing community should begin serious experimentation *concur*.
>
> Sperberg-McQueen und Huitfeldt (1998)

Aus einer inhaltlichen Perspektive betrachtet, wäre es also sehr sinnvoll diese erweiterte Möglichkeit von SGML nutzen zu können. Das größte Hindernis für den Einsatz des CONCUR-Features besteht jedoch darin, dass auch 15 Jahre nach der Verabschiedung des ISO-Standards praktisch keine Software existiert[21], die die Verarbeitung dieser speziellen Syntaxmöglichkeit von SGML unterstützte. Nahezu zwangsläufig ergab sich, dass diese Möglichkeit der elaborierten Dokumentstrukturierung nicht im Inventar der eXtensible Markup Language zu finden ist – einer Sprache, zu deren wichtigsten Designzielen auch eine einfache Implementierbarkeit gehört.

3.2 Direkte Verknüpfung der Auszeichnungsebenen durch Hypertext-Techniken

Charles Goldfarb verwarf nun nicht einfach nur die Möglichkeit der Verwendung von `CONCUR` zum Zweck der Markierung verschiedener logischer Hierarchien, sondern er schlug auch eine Alternative vor. Die Fortsetzung des oben begonnenen Goldfarb-Zitates lautet nämlich:

> Hypertext links are more appropriate for such applications (d. h., to create multiple logical views of a document; AW), and can be created easily by taking advantage of unique identifiers and external references.
>
> Goldfarb (1990:304)

In den folgenden Abschnitten sollen diese – mittels derartiger Hypertextkonstrukte – einfach erstellbaren Verknüpfungen von Dokumentteilen, und die sich dadurch potentiell modellierten, multiplen Hierarchien diskutiert werden.

---

[21] Die einzige (häufig genannte) Software, die SGML vollständig unterstützt, bildet die Produktfamilie Mark-It des Markup Technology Centre der belgischen Firma Sema.

3.2.1 Verweise auf eine vorhandene primäre Annotation

Vielfach wurde vorgeschlagen, eine primäre Annotationsebene zur Markierung der wesentlichen strukturellen Einheiten zu verwenden und zusätzliche Ebenen der Annotation mit dieser Ebene zu verknüpfen (vgl. u. a. Barnard et al., 1995, Sperberg-McQueen und Burnard, 1994). Die grundsätzliche Arbeitsweise ist in einer Beschreibung an einem linguistischen Beispiel sehr gut dargestellt (McKelvie et al. 2001). Diese Darstellung soll hier wiedergegeben werden, insbesondere auch weil in diesem neueren Modell der zusätzlichen Annotation konkurrierender Annotationsebenen, die derzeit aktuellen, d. h. XML-basierten, Auszeichnungstechnologien verwendet werden.

Der Beispielsatz ‚A cat sat on the mat' soll einerseits bezüglich einer einfachen Phrasenstruktur, andererseits bezüglich seiner (kontrastiven) Betonung annotiert werden. Die linguistische Annotation sieht folgendermaßen aus[22]:

```
<np><det>the</det><n>cat</n></np>
<vp><v>sat</v><pp><p>on</p>
<np><det>the</det><n>mat</n></np></pp>
</vp>
```

Die hierauf basierende Annotation, die dazu dient, die kontrastive Betonung zu repräsentieren, hat die Form:

```
<contrastive>the cat sat</contrastive> on the mat
```

Es ist zu sehen, dass eine einfache Vereinigung dieser Annotationen zu überlappenden Hierarchien führt. Die (u. a.) von MATE gewählte Lösung besteht aus mehreren Schritten. In einem ersten Schritt wird allen Elementen ein eindeutiger Identifikator zugewiesen. Dadurch wird ermöglicht, dass die Elemente potentiell als Verweisziele für zusätzliche Annotationsebenen dienen können.

Die Ergebnisannotation besteht dann aus verschiedenen Teilen: der Annotationsbasis und der zusätzlichen, mit ihnen durch Hypertextverknüpfungen verbundenen Annotationen:

[22] Das Vorhandensein des für vollständiges XML notwendigen, das gesamte Dokument umschließende Elementes (das Wurzelelement) ist hier ausgelassen. Der formale Status derartiger Annotationen wird in der Fußnote 32 auf Seite 66 kurz thematisiert.

```
<np><det id='x1'>the</det><n id='x2'>cat</n></np>
<vp>
<v id='x3'>sat</v><pp><p id='x4'>on</p>
<np><det id='x5'>the</det><n id='x6'>mat</n>
</np></pp></vp>
<contrastive href='id(x1)..id(x3)'>
```

Das Attribut `href` enthält als Wert die Zieladressen der Identifikatoren des ersten und des letzten Elementes des als ‚`contrastive`' markierten Bereiches. Dieses Attribut kann auch zum Verweis auf andere Dokumente verwendet werden, so dass die unterschiedlichen Auszeichnungen auch physikalisch separiert sein können.

Dieselbe Annotation kann in der Syntax des Konzeptes der ‚extended pointer' der TEI-Richtlinien (Sperberg-McQueen und Burnard, 1994:405ff.) durch die Attribute `from` und `to` spezifiziert werden. Eine entsprechende Darstellung der zusätzlichen Annotation mittels ‚extended pointer' kann folgendermaßen aussehen:

```
<contrastive>
<xptr doc='sys-anno.xml' from='ID (x1)' to='ID (x3)'>
</contrastive>
```

Die auch hier mögliche optionale Separierung der Annotationen erfolgt durch die Verwendung des Attributes `doc`.

Des Weiteren steht für denselben Zweck auch seit kurzem eine Verweistechnik zur Verfügung, die das XML-definierende Konsortium entwickeln ließ. Diese Technik wird als XPointer bezeichnet (DeRose et al. 2001). Eine zu den oben angegebenen MATE- und TEI-Verweisen analoge Darstellung, die als Verweissyntax XPointer verwendet, macht von einer Funktion `range` Gebrauch, die einen gesamten Bereich markiert.

```
...
<contrastive
xlink:href=„#xpointer(id(„x1“)/range-to(id(„x3“)))“/>
```

Es ist zu sehen, dass es eine Vielzahl von Möglichkeiten gibt, Verknüpfungen in XML vorzunehmen und diese zum Zwecke der Einfügung zusätzlicher Annotationsebenen zu verwenden. Neben diesen elaborierteren Möglichkeiten der Verknüpfung von Dokumentteilen gibt es auch noch die XML-inhärente Verweisfunktionalität mit den Attributtypen `ID` und `IDREF(S)`.

Die Verbindung einer zusätzlichen Auszeichnungsebene mit der primären Auszeichnungsebene durch Hypertexttechniken kann, wie gesehen, in vielfacher Weise erfolgen. Das Grundprinzip bleibt invariant:

- Eine primäre Annotationsebene wird erstellt.
- Die Elemente der primären Ebene werden mit ‚potentiellen' Hyperlinkzielen versehen.
- Die weiteren Annotationsebenen beinhalten Markup, das mittels Hyperlinks auf die primäre Ebene verweist.

Ein Problem dieses Ansatzes besteht darin, dass die potentiellen Beginn- und Endmarkierungen der Umgebungen der weiteren Annotationsebenen bereits in der Basisannotation vorhanden und mit Identifikatoren versehen sein müssen. Von dieser Situation kann im Normalfall nicht ausgegangen werden. Beispielsweise annotiert die im obigen Beispiel verwendete Basis die Wortarten. Die in der zweiten Hierarchie vorgenommene Markierung der Betonung ist jedoch keineswegs an diese Wortgrenzen gebunden, da auch der Fall eintreten kann, dass nur ein Teil eines Wortes, z.B. eine Silbe, kontrastiv betont ist. Entsprechend benötigt eine phonetisch orientierte Annotation eine detailliertere Basisannotation. Eine Lösung hierfür kann darin bestehen, eine Ebene als Basis zu wählen, die nicht weiter unterteilt werden kann. Jedoch gäbe es dann das Problem, dass geklärt werden müsste, welches diese Ebene sein könnte.

Ein weiterer Nachteil dieses Ansatzes besteht darin, dass sich – mit Ausnahme der Basisannotationsebene – die einzelnen Annotationsebenen nicht aus sich heraus erklären. So ist es im obigen Beispiel notwendig, bei der Betrachtung der Ebene der kontrastiven Markiertheit die Annotation der Wortarten mit zu betrachten.[23] Die Ebenen müssen darüber hinaus, wenn eine sinnvolle Verarbeitung vorgenommen werden soll, zusammengeführt werden. Dadurch kann beispielsweise die Aufgabe der farblichen Hervorhebung, der als kontrastiv-markierten Textteile, zu einem recht aufwändig zu lösenden Problem werden. Wäre die Markierung der Kontrastivität hingegen in die Basisannotationsebene integriert worden, wäre diese Aufgaben mit einfachsten texttechnologischen Mitteln zu lösen.

[23] Im Abschnitt 6.1.3 (ab S. 141) wird dies bei der genaueren Beschreibung von MATE nochmals angesprochen und konkretisiert.

3.2.2 Verweise auf Zeitachsen

In einem allgemeinen Sinn verstanden können die als Sprachdaten bezeichneten Einheiten sehr unterschiedlicher Natur sein. Es gibt Wörterbücher, Grammatiken, Datenbanken etc. die Informationen über sprachliche Einheiten zusammentragen und somit Sprachdaten bilden. In einem engeren Sinne können Sprachdaten jedoch auch als Sprachausschnitte verstanden werden, die in Korpora zusammengefasst sind. Eine Untereinheit dieser Korpora bilden Ausschnitte gesprochener Sprache. Diese Gruppe von Daten ist im Normalfall dadurch geprägt, dass es eine physikalisch repräsentierte primäre Datenebene – die Aufzeichnung der gesprochenen Sprache – gibt. Die weiteren Ebenen der Datenrepräsentation, seien es die Transkriptionen, seien es Analysen, können dann als sekundäre Datenebenen angesehen werden. Aus diesem Blickwinkel heraus betrachtet, ist es naheliegend, eine Verankerungsbasis herauszugreifen: Sie kann nur auf der Ebene der Primärdaten gesucht werden.

Bird und Libermann (2001) entwickeln einen formalen Rahmen zur Repräsentation sprachlicher Daten, die „Annotation Graphs“. Sie bilden ein mathematisches Modellierungsinstrument, um linguistische Annotationen vorzunehmen, die folgendermaßen beschrieben sind:

> In the simplest and commonest case, ‘linguistic annotation’ is an orthographic transcription of speech, time-aligned to an audio or video recording. Other central examples include morphological analysis, part-of-speech tagging and syntactic bracketing; phonetic segmentation and labeling; annotation of disfluencies, prosodic phrasing, intonation, gesture, and discourse structure; marking of coreference, ‘named entity’ tagging, and sense tagging; and phrase-level or word-level translations. Linguistic annotations may describe texts or recorded signals. Our focus will be on the latter, broadly construed to include any kind of audio, video or physiological recording, or any combination of these, for which we will use the cover term ‘linguistic signals’. However, our ideas also apply to the annotation of texts.
>
> Bird und Libermann (2001:25)

Aus dem letzten Satz des Zitats geht hervor, dass auch andere absolute Markierungen als die ‚timeline’, als Verankerungsbasis verwendet werden können. Die Primärdatenachse muss nicht notwendigerweise der die Dauer der Äußerungen festhaltende Zeitstrahl sein. Dies erlaubt die Anwendbarkeit dieses Ansatzes auch auf textuelle Daten, die nicht aus einer Transkription gesprochener Sprache resultieren (s. u.).

Ein Annotationsgraph erlaubt die Verknüpfung einer Menge von Knoten, die direkt oder indirekt mit einer ‚timeline' verbunden sind, mittels benannter Kanten, d. h. Sprachdaten werden so repräsentiert, dass eine primäre Datenebene und eine erweiterbare Menge von Annotationsebenen aufgebaut wird. Diese Ebenen sind miteinander mittels einer ‚timeline' verknüpft, deren Funktion folgendermaßen beschrieben wird:

> In our formalization of annotation graphs, the only thing that really matters about time references is that they define an ordering.
>
> Bird und Libermann (2001:46)

Die von Bird und Libermann erklärten Ziele des Formalismus können mittels der Kriterien *Einfachheit* sowie *Balance zwischen Generalisierbarkeit und Spezifik* ausgedrückt werden. Die ausführliche Formalisierung und die Anwendbarkeit der Annotationsgraphen auf eine immense Anzahl existierender Enkodierungen linguistischer Daten stieß in der linguistischen Forschung auf eine äußerst große und mehrheitlich positive Resonanz.

Die Wahl der Verknüpfungsbasis der Annotationsgraphen – die ‚timeline' – ist jedoch schon vorher als Option für eine möglichst exakte Transkription gesprochener Sprache vorgeschlagen worden. So findet sich in der Dokumentation zur DTD der *Text Encoding Initiative* folgender Vorschlag zur Verbindung bzw. Synchronisation verschiedener primärer Datenflüsse:

> For more complex kinds of alignment, involving possibly multiple synchronization points, an additional element is provided, known as a `<timeLine>`. This consists of a series of `<when>` elements, each representing a point in time, and bearing attributes which indicate its exact temporal position relative to other elements in the same timeline, in addition to the sequencing implied by its position within it.
>
> Sperberg-McQueen und Burnard (1994:314)

Die Notwendigkeit der Repräsentation des *Alignment* ergibt sich zum Beispiel bei der Transkription sich überlappender Redepassagen von mehreren Sprecher(inne)n oder auch bei Überlappung, wie sie bei der Wiedergabe der Gleichzeitigkeit sprachlicher und nichtsprachlicher Ereignisse (z. B. Gestik) auftritt. Es erscheint über diesen primären Zweck des Gebrauchs des Elementes `<timeline>` durchaus intendiert zu sein, es auch für weitere Ebenen der Annotation zu gebrauchen:

> The timeline mechanism is flexible and can handle *the various kinds of overlap in speech* far more precisely then existing transcription schemes.
>
> Johansson (1995: 156, Hervorhebung AW)

Auch wenn die Grundideen der Annotationsgraphen mit ihnen vergleichbare Vorläufer besitzen, stellen sie doch nicht nur im Sinne ihrer exakten Formalisierung und ihrer Anwendbarkeit auf quasi alle bisher verwendeten Transkriptionsschemata eine Besonderheit dar, sondern auch wegen einer bereits im Eingangszitat von Bird und Libermann angedeuteten weiteren Anwendungsmöglichkeit: Annotationsgraphen können im Prinzip auch auf textuelle Daten angewendet werden, die keine zeitliche Verankerung besitzen.

> We have focused on the case of audio or video recordings, where a time base is available as a natural way to anchor annotations. This role of time can obviously be reassigned to any other well-ordered single dimension. The most obvious case is that of character- or byte-offsets into an invariant text file. This is the principle used in the Tipster Architecture (Grishman, 1997), where all annotations are associated with stretches of an underlying text, identified via byte offsets into a fixed file.
>
> Bird und Libermann (2001:56)

Zusammenfassend bleibt festzuhalten, dass der Ansatz der Darstellung linguistischer Daten mit Annotationsgraphen derzeit eines der vielversprechendsten Modelle darstellt, insbesondere da ihre Anwendungsmöglichkeiten nicht auf Transkriptionen gesprochener Sprache beschränkt sind. Allerdings finden sich die Grundideen bereits in den wesentlich älteren TEI-Richtlinien wieder. Dort gibt es u. a. auch das Konzept der Zeitachse, welche sich im Element `<timeline>` niederschlägt.

Die Verwendung einer Zeit- bzw. einer anderen als primär definierten Datenachse als Referenzbasis bringt jedoch auch einige Nachteile mit sich. Einer der gravierendsten Mängel wurde in der Kurzbeschreibung zur Verwendung von `<timeline>` durch die TEI erwähnt:

> „It is a problem, however, that the coding becomes rather complex“.
>
> Johansson (1995:156)

Alle weiteren Probleme ließen sich auf dieses Komplexitätsproblem zurückführen, sie besitzen jedoch einen eigenen Stellenwert. So ist es möglich, dass der Beginn und das Ende einer linguistischen Einheit, z. B. eine Phrase oder ein Morph, nur durch Verweise auf die separate Ebene, die ‚timeline’, definiert wird. Dadurch ist es jedoch nicht mehr möglich, in einer einfachen Weise –

d. h. mit Standardsoftware – anzugeben, dass das geöffnete Morphem wieder geschlossen werden muss oder dass Morphe in Phrasen vorkommen können, jedoch Phrasen nicht in Morphen.[24]

Ein weiteres Problem besteht darin, dass eine einzelne Annotationsebene separat, d. h. ohne die Basisannotation, nicht verwendbar ist. Dieses Problem, mit einem Beispiel versehen bereits im Abschnitt 3.2.1 beschrieben, ist für die Annotationsgraphen noch zentraler, da auch indirekte Verweise auf die Zeitachse vorgesehen sind. Da eine Vielzahl von Querbezügen zwischen den Annotationsebenen existieren kann, müssen u. U. alle Ebenen betrachtet werden, damit alle ebenenbezogene Informationen verfügbar sind. Neben dem hohen Verarbeitungsaufwand ist es hierdurch de facto unmöglich, die Annotation nur weniger Ebenen – eine Art partieller Annotation – auszutauschen.

3.3 ‚Meilensteine'

Eine sehr einfache und nicht zuletzt deshalb häufig verwendete Option zur Markierung weiterer Hierarchieebenen in SGML- und XML-Dokumenten besteht in der Einfügung von zusätzlichen leeren Elementen, die ausschließlich dazu dienen, eine oder mehrere zusätzliche Strukturierungsebenen einzufügen.

Die am Beginn dieses Kapitels vorgestellte OHCO-These, die besagt, dass Texte eine „ordered hierarchy of content objects" bilden (vgl. S. 38), findet in SGML nur in den Elementen eine syntaktische Entsprechung, deren Inhaltsmodell nicht als leer definiert wurde.[25] In diesem Fall bilden die „content objects" Container, die Daten oder andere Container enthalten und selbst – mit

[24] Dies bedeutet de facto, dass aus den Möglichkeiten SGML-basierter Informationsmodellierung kein Nutzen gezogen werden kann. Dies gilt unabhängig davon, dass SGML als Annotationsformat der TEI und XML als Austauschformat der Annotation Graphs verwendet wird.

[25] Grundsätzlich besteht jedoch auch die Möglichkeit, Inhaltsdaten in leeren Elementen als Attributwert (z. B. als `CDATA`) einzufügen. Allerdings verstößt dieses Vorgehen gegen die etablierte Konvention, Primärdaten, d. h. Daten die potentiell direkt angezeigt oder gedruckt werden sollen, in Elementinhalten zu halten. Dieses Prinzip wird von nahezu allen etablierten DTDs befolgt, nicht zuletzt, weil die Missachtung dieses Grundsatzes zu praktischen Problemen führen kann. So erlaubt die Formatierungssprache CSS ihrer Basisdefinition (CSS level 1, vgl. Lie und Bos (1999)) nur die Anzeige von Element- nicht aber von Attributinhalten. (vgl. auch Sasaki und Witt 2001)

Ausnahme des Dokument- bzw. Wurzelelementes in anderen Containern enthalten sind. Leere Elemente erzeugen keine Container, sie markieren nur Punkte. Bei einer kritischen Betrachtung der in den häufigsten Dokumentgrammatiken verwendeten leeren Elemente fällt auf, dass in vielen Fällen ihr Verwendungszweck darin besteht, indirekt Umgebungen zu markieren. In der HTML-DTD wird z. B. ein Element `<br>` definiert, das die Position eines Zeilenumbruchs markiert. Zwischen zwei dieser Zeilenumbrüche befindet sich, falls dieses Element konsistent als Strukturierungsinstrument benutzt wird, der Inhalt der Zeilen.

Die Text Encoding Initiative gibt als einen weiteren von verschiedenen Auswegen aus der Restriktion nur jeweils eine Hierarchie in einem SGML-Dokument definieren zu dürfen, die Verwendung von leeren Elementen an, die als Meilensteine bezeichnet werden. Diese Meilensteine werden ähnlich wie die beschriebene Verwendung des HTML-Elementes
 als indirekte Umgebungsmarkierungen benutzt. Die TEI gibt einige allgemeine und einige spezielle Meilensteinelemente an.

Ein Beispiel für ein spezielles Meilensteinelement ist das zur Markierung der Seitenumbrüche (z. B. in Manuskripten) verwendete leere Element `<pb>`[26]. Die TEI-Richtlinien stellen für den selben Zweck auch ein Element mit einem nichtleeren Inhaltsmodell zur Verfügung – `<page>`.

Da die Paginierung nun aber ein zu den anderen Strukturierungseinheiten unabhängiges Ereignis ist, entstünden potentiell beim Einsatz des Elementes `<page>` überlappende Inhaltseinheiten, weshalb es nur bei der Nutzung der erweiterten SGML-Möglichkeit CONCUR verwendet werden soll (vgl. Abschnitt 3.1). Als alternativ hierzu kann die Verwendung des leeren Elementes `<br>` aufgefasst werden: alles zwischen zwei Seitenwechseln gehört zu einer Seite.

Das von der TEI vorgeschlagene allgemeine Element zur Notierung von derartigen Meilensteinen heißt `<milestone>`, (Sperberg-McQueen und Burnard, 1994:187f.), und ähnlich wie Meilensteine auch für die Markierung von Kilometern oder anderen Längeneinheiten in einer konkretisierten Ausprägung gewissermaßen eingeschränkt verwendet werden können, kann die Form der Verwendung des allgemeinen Elements `<milestone>` durch die Angabe eines Wertes für ein Attribut `unit` spezifiziert werden. Es ist also

[26] `<pb>` steht für page break.

auch möglich, statt des Elementes
 auch das allgemeine Element zur Markierung von Meilensteinen, nämlich:

```
<milestone unit="page">
```

zu verwenden, wobei allerdings die Semantik dieser Element-Attribut-Wert-Spezifikation nicht in den TEI-Richtlinien definiert ist.

Einen Schritt weiter als die einfache Markierung von Meilensteinen gehen Barnard et al. (1995). Sie führen in Analogie zu <milestone> die Elemente <action>, <move>, <gesture> etc. ein, um parallel zum Text ablaufende nichtsprachliche Ereignisse zu annotieren. Diese Elemente können in einer direkteren Form zur Markierung von Umgebungen gebraucht werden, da sie nicht nur eine mit einer Beschreibung versehene Marke bilden, sondern erlauben, den Beginn und das Ende einer Umgebung auszuzeichnen. Dies geschieht, indem ein Attribut eingeführt wird, dessen Wert entweder start oder end sein kann. Sie bilden somit das Analogon zu den speziell hierfür vorgesehenen Markierungen des Umgebungsbeginns (<...>) bzw. des Umgebungsendes (</...>).

Die sich durch die Syntaxdefinitionen von SGML ergebende Paarigkeit der Beginn- und Endmarken muss bei der Nutzung der leeren Elemente anderweitig hergestellt werden. Auch hier liegt die Lösung in einer weiteren Attributierung, eine der Marken, meist die als start gekennzeichnete, erhält ein Attribut, dessen Wert vom Typ ID sein muss, das Gegenstück hierzu, also meist das Umgebungsende, referiert hierauf mittels eines Attributs vom Typ IDREF. Es ist also möglich, mit einem derartigen Gebrauch leerer Elemente Umgebungen nachzubilden und innerhalb des Standardfunktionsumfanges von SGML, d. h. insbesondere auch innerhalb von XML, multiple Hierarchien zu annotieren – und dies nicht separat, sondern an den Stellen, an denen sie auftreten.

Die Verwendung der Meilensteine bringt jedoch auch eine Vielzahl von Nachteilen mit sich. So benennen Durand et al. (1996) das bereits in vergangenen Abschnitt angesprochene Problem der Paarigkeit und die damit verbundenen Schwierigkeiten. So kann nicht (mit den Mitteln von SGML) überprüft werden, ob es für die jeweiligen Startmarken auch Endmarken gibt oder ob die Startmarke vor der Endmarke auftritt. Darüber hinaus ist es nicht möglich, Beziehungen zu den anderen Elementen des Textes zu spezifizieren, da es in der Natur leerer Elemente liegt, keine Inhaltsmodelle zu besitzen, die

restringiert werden können. Weitere Nachteile werden ausführlich in der Diskussion (Abschnitt 3.8 ab Seite 71) ausgeführt.

3.4 Fragmentierung von Elementen

Ebenfalls von der TEI stammt der folgende, gleichfalls sehr einfache Problemlösungsvorschlag. Die von den Umgebungsüberschneidungen betroffenen Elemente werden zur Annotation fragmentarischer Umgebungen verwendet. Dies bedeutet, dass vollständige Umgebungen, z. B. ein Zitat oder ein Satz dann in mehrere Umgebungen aufgeteilt werden, wenn andere, gleichzeitig wirksame Umgebungen, die schon vor dem Beginn der entsprechenden Umgebung geöffnet waren, beendet werden. Es sei als Beispiel eine Annotation des folgenden Ausschnittes aus Schillers Ballade „Die Bürgschaft“ gewählt:

...
»Was wollt ihr?« ruft er, für Schrecken bleich,
»Ich habe nichts als mein Leben,
Das muß ich dem Könige geben!«
Und entreißt die Keule dem nächsten gleich:
»Um des Freundes willen erbarmet euch!«
Und drei mit gewaltigen Streichen
Erlegt er, die andern entweichen.
...

In einer Beispielauszeichnung wird die wörtliche Rede mit <q> und die Aufteilung in Zeilen mit <l> markiert:

```
...
<l><q>Was wollt ihr?</q>ruft er, für Schrecken bleich,</l>
<l><q>Ich habe nichts als mein Leben,</q></l>
<l><q>Das muß ich dem Könige geben!</q></l>
<l>Und entreißt die Keule dem  ...
```

Die Beendigung des ersten Zitats – Was wollt ihr? – erscheint durchaus sinnvoll, da anschließend ein beschreibender Diskurs folgt, wohingegen die Beendigung der Umgebung <q> hinter dem Wort ‚Leben’ nur dem Umstand zuzuschreiben ist, dass an dieser Stelle das Ende der Zeilen (<l>) markiert werden muss. Wenn mit derartigen Fragmentierungstechniken gearbeitet wird, werden die vollständigen Inhalte von Umgebungen bei Bedarf auf verschiedene Umgebungen verteilt. Problematisch ist hierbei, dass nicht erkennbar ist, warum eine Umgebung beendet wurde, bzw. ob sie vollständig ist

oder nicht. Dies kann jedoch durch die Einführung von Attributen, die die Verkettung zusammenhängender Umgebungen markieren, vermerkt werden.

```
...<l><q status='complete'>Was wollt ihr?</q>ruft er, für
Schrecken bleich,</l>
<l><q id='n1' status='partial' next='n2'>Ich habe nichts
als mein Leben,</q></l>
<l><q id='n2' status='partial' prev='n1'>Das muß ich dem
Könige geben!</q></l><l>Und entreißt die Keule dem  ...
```

Das Attribut `status` vermerkt den Grad der Vollständigkeit und die Attribute `next` und `prev`, deren Werte Verweise (`IDREFS`) sein können, erlauben die Rekonstruktion der gesamten Umgebung mit einfachen Mitteln. Allerdings kann mit den Mitteln von SGML nicht überprüft werden, ob alle Teile einer als unvollständig markierten Umgebung vorhanden sind.

Eine derartige Annotation kann jedoch aus theoretischer Perspektive auch dann sinnvoll sein, wenn in formaler Hinsicht keinerlei Notwendigkeit für ein derartiges Vorgehen besteht. Auch hierfür soll ein Beispiel – nämlich die Annotation der aus der traditionellen Linguistik stammenden funktionalen Einheit ‚Prädikat' – gewählt werden. In deutschen Sätzen tritt sehr häufig ein komplexes Prädikat auf. In dem Satz „Vor der ersten Inbetriebnahme müssen unbedingt alle Teile der Transportsicherung entfernt sein" befindet sich das komplexe Prädikat „müssen entfernt sein". Eine sehr einfache, XML-konforme Möglichkeit der Annotation bestünde nun in der folgenden[27]:

```
<s><pp>Vor der ersten Inbetriebnahme</pp>
   <pred>müssen<adv>unbedingt</adv>
               <np>alle Teile der Transportsicherung</np>
         entfernt sein</pred></s>
```

Es finden nur ineinander verschachtelte Umgebungen Anwendung, so dass keine syntaktischen Verstöße gegen XML vorkommen. Nichtsdestotrotz kann eine derartige Annotation nicht befriedigen, da das Adverb ‚unbedingt' und der Nominalkomplex ‚alle Teile der Transportsicherung' nicht Teil des komplexen Prädikates sind, wie die Annotation suggeriert.

Die Verwendung von Fragmenten würde diese Interpretation nicht erlauben:

[27] Bedienungsanleitung „AEG Öko_Lavamat 6200, 9200, ..."

```
<s><pp>Vor der ersten Inbetriebnahme</pp>
   <pred type='complex' id='n1' status='partial'
         next='n2'>müssen</pred>
   <adv>unbedingt</adv>
   <np>alle Teile der Transportsicherung</np>
   <pred type='complex' id='n2' status='partial'
         prev='n1'>entfernt sein</pred></s>
```

Die Nutzung der Fragmentierungstechnik bietet, wie zu erkennen, vielfältige Anwendungsmöglichkeiten.

Trotz der bereits oben oder in der später folgenden Diskussion erwähnten Nachteile besitzt dieser Ansatz diverse Vorteile, insbesondere gegenüber der Meilensteinlösung. Hierbei ist zuerst die Verwendung von Umgebungen zu nennen, da diese die Voraussetzung für den Aufbau von Hierarchien bilden.

3.5 Implizite Verknüpfung der Auszeichnungsebenen

Das bereits oben erwähnte MATE-Projekt favorisiert die Verwendung der Verknüpfung von Informationseinheiten, um Annotationen auf verschiedenen (linguistischen) Beschreibungsebenen vornehmen zu können. In einer im Rahmen dieses Projektes publizierten Studie werden die theoretischen Aspekte hierzu diskutiert (Mengel und Heid 1998). Es werden vier prinzipielle Möglichkeiten der ebenenübergreifenden Auszeichnung thematisiert, wobei zwei dieser Optionen bzw. Infrastrukturen der Verknüpfung dergestalt aufgebaut sind, dass sie bei einer oberflächlichen Betrachtungsweise nicht als Verknüpfung erkennbar sind:

- ein Fehlen einer Infrastruktur zur Ebenenverknüpfung und
- eine theorieimmanente Ebenenverknüpfung.

Der Verzicht auf die Angabe von Beziehungen zwischen den Ebenen wird explizit als ein Infrastrukturtyp aufgefasst:

> No infrastructure refers to all those corpora that possibly allow retrieval and identification of cross-level phenomena, but do not provide explicit infrastructure for this purpose. Thus, there might be a possibility of investigating cross-level phenomena because the data are described on different layers although no special effort has been made to facilitate this. However, in this case, cross-level queries are neither intended nor especially supported.
>
> Mengel und Heid (1998)

Dieser Weg stellt also, auch wenn das MATE-Projekt ihn nicht weiter verfolgt hat, eine prinzipielle Möglichkeit der Annotation von Phänomenen dar, die auf unterschiedlichen Beschreibungsebenen angesiedelt sind. Welcher Art die Beziehungen sind und wie diese herausdestilliert werden können, ist die Aufgabe weiterer Verarbeitungsprozesse. Aus dem Zitat geht hervor, dass dieser Ansatz insbesondere deshalb kritisch betrachtet wird, da der Verzicht auf eine explizite Datenaufbereitung dazu führt, dass die Exploration der ebenenübergreifenden Phänomene durch Anwendungen nicht intendiert oder gar unterstützt wird.

Es liegt jedoch keineswegs auf der Hand, dass es sich dabei um einen Nachteil handelt. Vielmehr kann auch so argumentiert werden, dass genau dieses Fehlen Vorteile mit sich bringen kann. Ein gewisser Vorteil, der jedoch rein pragmatischer Natur ist, besteht darin, dass der Aufwand für die Erstellung annotierter Korpora geringer ist, wenn auf die explizite Angabe der ebenenübergreifenden Beziehungen verzichtet wird. Ein eher auf der theoretischen Seite liegender Vorteil begründet sich aus der Betrachtungsweise der Korpora. Implizite Beziehungen müssen gesucht bzw. herausgearbeitet werden, was dazu führen kann, dass Beziehungen gefunden werden, die für die Personen, die das Korpus erstellt haben, nicht relevant waren oder derer sie sich nicht bewusst wurden. Eine explizite Annotation kann den Blick hierfür verstellen, da das Korpus meist genau so genutzt wird, wie es vorliegt. Dies führt meist dazu, dass in diesen Korpora nicht mehr nach ‚verborgenen' Informationen gesucht wird, obwohl diese natürlich auch dort vorhanden sind. In dieser Hinsicht ergeben sich Parallelen zu der in dieser Arbeit favorisierten und im Abschnitt 3.7 vorgestellten separaten Annotation der linguistischen Beschreibungsebenen. Derartige impliziten Verknüpfungen bleiben aber hinter dem neuen Ansatz zurück, da eine fehlende Infrastruktur keine Antworten auf die Probleme gegensätzlicher Hierarchien bietet. Es muss somit auch hier auf die bereits geschilderten ‚workarounds' zurückgegriffen werden.

Die zweite Form der impliziten Verknüpfung von Auszeichnungsebenen, die von Mengel und Heid (1998) ‚theorieinhärente Infrastruktur' genannt wird, findet sich in einer Vielzahl von Annotationsschemata. Als Beispiel nennen sie die Annotation von „Einheiten unterschiedlicher syntaktischer Komplexität: Wörter, Phrasen und Sätze", wobei die Satzgrenzen auch die Phrasengrenzen und die Wortgrenzen bilden. Auch diese Infrastruktur wurde vom MATE-Projekt nicht weiter verfolgt.

Beiden Infrastrukturen ist gemein, dass durch sie die Komplikation nicht ausgeräumt werden kann, die entsteht, wenn verschiedene Hierarchien in die Annotation inkorporiert werden müssen. Für die Vielzahl von Daten, die ohnehin der OHCO-These genügen, stellt die implizite (Nicht-)Angabe von Beziehungen zwischen den Annotationsebenen jedoch eine Option dar, einen Mehrwert aus derartigen Korpora zu ziehen.

Sollen derartige implizite Bezüge in explizite Verknüpfungen verwandelt werden, stellen SGML-Architekturen Techniken bereit, die u. a. für derartige Zwecke genutzt werden können.

3.6 SGML-Architekturen

Im Anhang zur zweiten, revidierten Auflage des HyTime-Standards (ISO 10744, 1997) werden die sogenannten SGML extended facilities definiert. Damit werden die Möglichkeiten zur Verwendung von SGML erweitert, so dass Konstrukte geschaffen werden können, deren Ausdrucksmächtigkeit über die der im SGML-Standard befindlichen Konstrukte hinausgeht.

Eine dieser extended facilities – festgehalten und standardisiert im HyTime-Anhang A.3: Architectural Form Definition Requirements (AFDR) – dient der Angabe von Beziehungen zwischen verschiedenen DTDs.

Jede beliebige SGML-DTD kann zu einer Meta-DTD erklärt werden. Eine weitere DTD, bezeichnet als Client-DTD, wird mit Informationen angereichert, die die als architektonisch bezeichneten Beziehungen der in ihr definierten Elemente und Attribute zu den von der Meta-DTD definierten Strukturierungseinheiten beschreiben. Sogenannte architectural engines können eine entsprechend der Client-DTD annotierte Dokumentinstanz dergestalt verarbeiten, dass die architektonischen Beziehungen ausgewertet werden. Das Ergebnis dieser Verarbeitung ist dann ein abgeleitetes SGML-Dokument. Das abgeleitete Dokument besteht aus der durch die architectural engine erzeugte Dokumentinstanz und der Dokumentgrammatik, die vorher zur Meta-DTD erklärt wurde.

Da beliebige DTDs zu Client-DTDs erklärt werden können, ist es auch möglich, dass eine Meta-DTD ihrerseits wiederum die Client-DTD für eine weitere Meta-DTD bildet. Dies erlaubt es, eine architektonische Verarbeitung höherer Ebene vorzunehmen. Es können dabei Ableitungsketten erstellt werden. Es ist auch möglich, einer Client-DTD mehrere Meta-DTDs zuzuordnen, wobei jedoch die Ableitung immer nur nach einer Meta-DTD erfolgt. Somit entsteht bei einer derartigen Verarbeitung keine architektonische Verarbeitung höherer Stufe. Lobin (2000:179f.) konnte jedoch zeigen, dass derartige Komplexitätsstufen durchaus relevant sein können: durch die Kombination mit der höherstufigen architektonische Verarbeitung entstehen netzartige Strukturen, sogenannte DTD-Netze.

Lobin (2000) hebt besonders den konzeptuellen Unterschied zwischen der klassischen Dokumentstrukturierung und den Strukturierungsmöglichkeiten durch Architekturen hervor. Die klassische Strukturierung, die Restringierung der Ausprägung einer Dokumentinstanz durch die DTD, bezeichnet er als primäre Informationsmodellierung. Werden mittels architektonischer Formen jedoch Beziehungen zwischen den Dokumentgrammatiken definiert, spricht er von der sekundären Strukturierung bzw. Informationsmodellierung.

Eine einfache Form von architektonischer Beziehung ergibt sich dann, wenn Element- und/oder Attributnamen umbenannt werden sollen. Die Struktur des Dokumentes bleibt bei einer derartigen architektonischen Verarbeitung unverändert. So kann die deutschsprachige Annotation linguistischer Informationen in eine englischsprachige transferiert werden.

```
Original:
<Artikeltitel><wort wortart='n'>Streit</wort>
<pp>um die Sprache</pp></Artikeltitel>
Ableitung:
<title><word part_of_speech='n'>Streit</word>
<pp>um die Sprache</pp></title>
```

Es ist auch möglich, das Auftreten bestimmter Attributwerte abzufragen und ihr Auftreten dann, in einer strikt vorgeschriebenen Weise, zur Generierung der anderen im abgeleiteten Dokument vorhandenen neuen Attributwerte zu verwenden.

```
Original:
<wort wortart='Substantiv'  lexem='Baum'>Bäume</wort>
Ableitung:
<word part_of_speech='noun' lexem='tree'>Bäume</wort>
```

Die Dokumentstruktur eines Basisdokuments und des von ihm architektonisch abgeleiteten Dokuments kann sich jedoch auch in einem gewissen Umfang unterscheiden. So kann das Basisdokument gefiltert werden, d. h. es ist möglich, Ableitungen zu erstellen, bei denen bestimmte Attribute, Attributwerte, Elemente oder Elementinhalte nicht in das Zieldokument übernommen werden.

```
Original:
<wort wortart='Substantiv' lexem='Baum'>Bäume</wort>
Ableitung:
<wort lexem='Baum'/>
```

Als letzte und weitreichendste Möglichkeit der Dokumentmanipulation kann mittels Architekturen der textuelle Elementinhalt in Attributwerte überführt werden und umgekehrt. So zeigt Lobin (1999b) wie linguistisch annotierte Satzinhalte mittels solcher Ableitungen zum Wert eines Attribute des Elternelementes werden. Wird diese Möglichkeit gemeinsam mit anderen – oben vorgestellten – Konstrukten verwendet, können bereits relativ umfangreiche Transformationen durchgeführt werde. Wird z. B. der (Unter-)titel eines Buch mit syntaktischen Annotationen versehen, kann die folgende SGML-Instanz entstehen:

```
<title><s type='ellipsis-verb'>
    <np><w>Unsere</w><sp> </sp><w>Gesellschaft</w></np>
    <sp> </sp>
    <pp><w>auf</w><sp> </sp><w>dem</w>
        <sp> </sp><w>Weg</w></pp>
    <sp> </sp>
    <pp><w>zur</w><sp> </sp><w>digitalen</w>
        <sp> </sp><w>Kultur</w></pp>
    <punctuation>.</punctuation>
</s></title>
```

Durch die Definition einer Meta-DTD und der Angabe architektonischer Formen kann eine Ableitung erstellt werden, bei der der gesamte textuelle Elementinhalt als Attributwert ausgewählter übergeordneter Elemente verwendet wird, wobei eine Duplizierung der Textausschnitte erfolgt. Erreicht wird durch eine derartige Ableitung eine noch stringentere Trennung des Inhaltes (d. h.

des Textes) von der Form (also der Strukturierung mittels MarkUp). Des Weiteren können Inhalte, seien es Textteile, Attribute oder Annotationen, daran gehindert werden, in der Ableitung aufzutreten. Dadurch ergibt sich eine Filterungsfunktionalität.

Eine komplexe, verschiedene Möglichkeiten der architektonischen Verarbeitung nutzende Architekturdefinition erlaubt es, aus dem oben annotierten Buchtitel folgendes Dokument abzuleiten[28]:

```
   <titel Inhalt=
"Unsere Gesellschaft auf dem Weg zur digitalen Kultur.">
     <satzstruktur typ="ellipsis-verb">
       <np Inhalt="Unsere Gesellschaft"/>
       <pp Inhalt="auf dem Weg"/>
       <pp Inhalt="zur digitalen Kultur"/>
     </satzstruktur></titel>
```

Für die Einfügung des textuellen Inhaltes in die Attributwerte ist es notwendig, explizite Trennzeichen, hier die im Basisdokument als `<sp>` annotierten Leerzeichen, anzugeben. Es ist nicht möglich, die Markierung des Endes (bestimmter) Umgebungen dafür zu nutzen, Trennzeichen automatisch einzufügen.[29] Damit bleibt der Weg für eine alternative, naheliegendere Ausgangsannotation der folgenden Form versperrt:

```
<title><s><np><w>Unsere</w><w>Gesellschaft</w></np>
          <pp><w>auf</w><w>dem</w><w>Weg</w></pp>
          <pp><w>zur</w><w>digitalen</w><w>Kultur</w></pp>
          <punctuation>.</punctuation></s></title>
```

Die auf dieser Grundlage erzeugbare Ableitung, die textuelle Elementinhalte in Attributwerte überführt, würde als Resultat keine der zwischen den Worten implizierten Leerzeichen enthalten, d. h. z. B.:

---

[28] Im Anhang A.1 finden sich die Basis- und die Meta-DTD.

[29] Dass der Anspruch nach einer Einteilung in trennzeichenevozierende und sonstige Elemente nicht so abwegig ist, zeigt z. B. die gängige Interpretation der HTML-DTD. Dort wirkt das Auftreten bestimmter Elemente, der sogenannten Blockelemente, z. B. `<H1>` oder `<LI>`, textdatenseparierend. Andere nichtzeilenumbrechende Elemente, z. B. `<em>`, bewirken keine Einfügung von Trennzeichen. In Analogie zu diesen zeilenseparierenden HTML-Elementen wäre es auch vorstellbar, mittels Architekturen wortseparierende und sonstige Elemente zu definieren. Da dies jedoch nicht vorgesehen ist, müssen in dem Beispiel die Leerzeichen explizit eingefügt werden.

```
<titel
Inhalt="UnsereGesellschaftaufdemWegzurdigitalenKultur."/>
```

Die Leerzeichen können also nicht durch eine architektonische Verarbeitung generiert werden. Diese Beschränkung der Anwendungs- und Verarbeitungsmöglichkeiten der Architekturen lässt sich generalisieren – durch architektonische Ableitungen lassen sich keine neuen Elemente generieren.[30]

Bei der Betrachtung von SGML-Architekturen bleibt zu fragen, welcherart die Beziehungen zwischen Basisdokumenten und Ableitung sein können. Auffällig ist, dass sich die Meta-DTD und die Client-DTD erheblich unterscheiden können.

Durch die Ableitung einer nach einer Basis-DTD annotierten Dokumentinstanz kann kein im Ausgangsdokument nicht vorhandener Inhalt erzeugt werden. Auf der Ebene der Dateninhalte kann neben der simplen Übernahme und der Filterung (d. h. Löschung) von Dateninhalten nur erreicht werden, Duplikationen vorhandener Daten zu generieren. Auf der Ebene der Annotation ist es möglich, neue Attribute und Attributwerte zu generieren, z. B. auch aus Elementinhalten.

In einer strukturellen Betrachtungsweise lässt sich die Unterschiedlichkeit zwischen Ausgangsdokument und abgeleitetem Dokument genauer klassifizieren. Dabei werden die Grenzen dieses Mechanismus deutlich. Die architektonischen Verarbeitungen der SGML-Elemente lassen sich in zwei Gruppen kategorisieren: Elementumbenennung und Elementeliminierung. Diese Betrachtungsweise wird deutlicher, wenn hierfür die mathematische Modellierung der Dokumentinstanzen herangezogen wird.

---

[30] Die Erzeugung von SGML-Elementen ist auch durch die Verwendung anderer innerhalb von SGML definierter Konstrukte nur in sehr wenigen Fällen zu erreichen. Ein in SGML – jedoch nicht in XML – für diese Zwecke verwendbares Konstrukt bildet die Möglichkeit `OMIT YES`. (vgl. Abschnitt 2.1.1) Wie schon im Abschnitt „Inferenz von Elementbezeichnern“ (ab S. 30) beschrieben, können bestimmte Markierungen trotz dieser grundsätzlichen Möglichkeit nicht weggelassen werden, z. B. wurde den Elementen, deren Inhaltsmodelle als leer (`EMPTY`) erklärt sind, von den Autoren des Standards explizit diese Möglichkeit genommen. Es wäre auch nicht möglich das oben zur Auszeichnung von Leerzeichen verwendete Element in der DTD als auslassbar zu deklarieren (`<!ELEMENT sp O O (#PCDATA)>`) und die Markierungen beim Auftreten von Leerzeichen einzufügen (z. B. `<w>auf</w> <w>dem</w>` → `<w>auf</w><sp> </sp><w>dem</w>`). Dies ist nicht möglich, da in SGML Leerzeichen mehrdeutig verwendet werden, einerseits als Daten, andererseits als Trennzeichen.

Werden in dem Dokument keine ID-IDREF-Referenzen verwendet, kann es mit einem geordneten Baum beschrieben werden. In diesem Fall gibt es für jeden Knoten im Baum genau einen Pfad zum Wurzelknoten. Die Anzahl der zwischen dem Elementknoten und der Wurzel liegenden Kanten ist die Verschachtelungstiefe des betreffenden Elementes. Für jedes Element im abgeleiteten Dokument gilt: Die Verschachtelungstiefe des Elementes ist kleiner oder gleich der Verschachtelungstiefe des Elementes des Ausgangsdokumentes, aus dem es abgeleitet wurde. Die Gleichheit der Verschachtelungstiefe ist z. B. bei der trivialen Ableitung eines identischen Dokumentes zu beobachten.

Die Verringerung der Baumtiefe soll anhand einer zweiten Architektur der oben wiedergegebenen Phrasenannotation illustriert werden:

```
<Wortannotation><w
                  >Unsere</w> <w>Gesellschaft</w> <w
                  >auf</w> <w>dem</w> <w>Weg</w> <w
                  >zur</w> <w>digitalen</w> <w>Kultur</w
>.</Wortannotation>
```

Es ist zu sehen, dass die Ebene der Phrasenannotation getilgt wurde und sich dadurch die Verschachtelungstiefe des Dokumentes um eine Ebene verringert.

Eine Ableitung, deren Baumrepräsentation eine größere Tiefe als die des Ausgangsdokuments besitzt, kann durch eine architektonische Verarbeitung nicht erstellt werden. Dafür wäre es entweder notwendig, dass neue Containerelemente generiert werden, die keine Entsprechung auf der Elementebene des Ausgangsdokumentes besitzen oder dass Transformationen angestoßen werden, die Verschiebungen von Subbäumen erlauben. Beides ist mit den Mitteln der SGML-Architekturen nicht möglich.

Die Limitierungen gehen jedoch noch weiter. Da nur gesamte Umgebungen miteinander in architektonische Beziehungen zueinander gesetzt werden können, ist es auch nicht möglich, bestimmte Anordnungen mehrerer Elemente (sogenannte Elementcluster) auf bestimmte Elemente zu beziehen. Beispielsweise kann durch eine Architektur zwar problemlos ausgedrückt werden, dass ein als <en> annotierter Eigenname auf das Element <np> bezogen werden soll (`<en>john</en> sang`), jedoch kann nicht ausgedrückt werden, das die Abfolge der mit `<det>` und `<n>` annotierten Wörter (`<det>der<det><n>mann</n>`) auf das Element `<np>` abgebildet werden soll, obgleich dadurch die Tiefe des Baumes nicht vergrößert würde.

3.7 Separate Annotation

Eine bisher noch äußerst selten betrachtete Möglichkeit zur Dokumentannotation besteht darin, die einzelnen Annotationsebenen vollständig zu trennen. Einzig die TEI-Guidelines erwähnen diese prinzipielle Vorgehensweise, führen sie aber nicht weiter aus (Sperberg-McQueen und Burnard, 1994:755f.). Nachfolgend soll diese Option ausgearbeitet werden, wobei Fragen der softwaretechnischen Umsetzung dieses Ansatzes anfangs ausgeblendet werden. Dies führt dazu, dass das einfachste System zu Redundanzen innerhalb der Daten führt. Die technischen Realisierungen der separaten Annotationen können jedoch dazu beitragen, das entstehende Datenvolumen erheblich zu reduzieren.

Das Grundprinzip der separaten Annotation ist denkbar einfach: Jede Annotationsebene wird separat annotiert. Hierbei werden die Primärdaten entsprechend der Anzahl der Annotationsebenen dupliziert. Hierfür muss zuerst geklärt werden, welche Daten der Terminus Primärdaten bezeichnet. Die Begriffsdefinition stuft die Primärdaten unabhängig von der verwendeten Schemasprache ein.

> PRIMÄRDATEN EINES XML-DOKUMENTS
>
> *Für ein wohlgeformtes XML-Dokument X wird eine DTD G erstellt. Wird X die DTD G zugeordnet entsteht das gültige Dokument X'. Die Menge aller Daten D die in der DTD G von X' als* `#PCDATA` *definiert wurden, werden als Primärdaten bezeichnet.*

Informeller ausgedrückt, zählen all die Daten zu den Primärdaten, die nicht zwischen einem MarkUp-Open-Delimiter (in XML immer <) und dem passenden MarkUp-Close-Delimiter (>) stehen. Durch eine derartige Auffassung von Primärdaten ist es nicht notwendigerweise so, dass Inhaltsdaten, z. B. Sprachdaten, als Primärdaten repräsentiert werden. Dies ergibt sich daraus, dass alle Inhaltsdaten auch als Attributwerte repräsentiert werden können – es ist sogar mit SGML-Architekturen möglich, diese verschiedenen Datenarten ineinander zu überführen.

Dies erfordert und ermöglicht es bei der Definition der Struktur der Dokumentklasse, eine weitere wichtige qualitative relevante Klassifikation der textuellen Daten vorzunehmen. Es muss entschieden werden, ob sie als Primärdaten eingestuft werden oder nicht. Die Primärdaten müssen als Dateninhalt von Elementen (`PCDATA`) definiert werden – und das in allen Annotationsebenen.

Nach der Klärung des Begriffs kann jetzt die Funktionsweise dieses Ansatz ausgeführt werden.

Während klassischerweise immer versucht wird, alle zu annotierenden Daten in einer logischen SGML- bzw. XML-Instanz aufzunehmen, wird hier davon ausgegangen, die Informationen auf mehrere, möglicherweise sehr viele, logische Dateien zu verteilen, eine Vorgangsweise, die einen fundamentalen Wechsel der Herangehensweise darstellt. Dies erscheint vermutlich überraschend und bedarf einer Rechtfertigung.

Die zugrundeliegende Idee besteht darin, dass es nicht notwendig ist, elaborierte Verknüpfungsmechanismen zu entwickeln und die entsprechenden Hyperlinks in die Annotationen einzubauen, sondern die annotierten Primärdaten als Basis der Verknüpfung zu verwenden. Da die annotierten Texte jeweils Kopien der Ausgangsdaten sind, wird für jede der separaten Annotationen derselbe Text verwendet. Dadurch können die Primärdaten als ‚Ermöglicher' oder ‚Enabler' der Verknüpfung aufgefasst werden.

Als Beispiel soll die Annotation eines Satzes aus einer Bedienungsanleitung[31] dienen. Der Satz „die Aufnahme stoppt und das Deck schaltet sich aus" wird bezüglich verschiedener Einheiten annotiert.[32]

Sätze (<s>):

```
<s><s>die Aufnahme stoppt</s> und <s
>das Deck schaltet sich aus</s></s>
```

Nominalphrasen (<np>):

```
<np>die Aufnahme</np> stoppt und <np>das Deck</np
> schaltet sich aus
```

Syntaktische Funktionen Subjekt (<subj>):

```
<subj>die Aufnahme</subj> stoppt und <subj
>das Deck</subj> schaltet sich aus
```

[31] Sony MiniDisk Deck – MDS-JB920

[32] Auf die Angabe des Kontextes bis hin zum Wurzelelement wurde verzichtet. Dies hat zur Folge, dass es sich bei den angegebenen separaten Beispielen i. d. R. nicht um wohlgeformte XML-Dokumente handelt. Ihr formaler Status entspricht den ‘well-balanced’ XML-Fragments. Die Definition, die in der W3C-Empfehlung „XML-Fragment Interchange" (Grosso und Veillard, 2001) zu finden ist, besagt, dass XML-Fragmente dann als well-balanced bezeichnet werden, wenn (1) alle geöffneten Umgebungen auch geschlossen worden sind und (2) die Umgebungsgrenzen sich nicht überlappen. Der wichtigste Unterschied zu wohlgeformten XML-Dokumenten besteht darin, dass kein Wurzelelement benötigt wird.

Verbalphrasen (<vp>):

```
die Aufnahme <vp>stoppt</vp> und das Deck <vp
>schaltet sich aus</vp>
```

Silben (<syll>):

```
<syll>die</syll> <syll>Auf</syll><syll>nah</syll><syll
>me</syll> <syll>stoppt</syll> <syll>und</syll> <syll
>das</syll> <syll>Deck</syll> <syll>schal</syll><syll
>tet</syll> <syll>sich</syll> <syll>aus</syll>
```

Morphe (<m>):

```
<m>die</m> <m>Auf</m><m>nahm</m><m>e</m> <m>stopp</m
><m>t</m> <m>und</m> <m>das</m> <m>Deck</m
> <m>schalt</m><m>et</m> <m>sich</m> <m>aus</m>
```

(vollständige) Wörter (<w>):

```
<w>die</w> <w>Aufnahme</w> <w>stoppt</w> <w>und</w
> <w>das</w> <w>Deck</w> schaltet <w>sich</w> aus
```

Es fällt auf, dass jede einzelne dieser unverbundenen Annotationen Gemeinsamkeiten und Unterschiede zu den anderen Einheiten besitzt. Systematischer betrachtet lassen sich die Daten einer jeden Annotation folgendermaßen einteilen: sie sind entweder Bestandteil des annotierten Textes oder sie gehören zu den annotierenden Daten.[33]

Die annotierenden Daten bestehen hier ausschließlich aus XML-Elementen, die nicht durch Attribute näher spezifiziert wurden. Die verwendeten XML-Elemente und ihre Einbettung ineinander unterscheidet die einzelnen Beispiele. Hingegen sind die annotierten Daten, hier also der textuelle Inhalt „die

[33] Dies ist nicht trivialerweise der Fall, d. h. diese starre Unterscheidung gilt nicht notwendigerweise für alle Daten innerhalb von SGML-Dokumenten. So können Attributwerte, insbesondere wenn sie aus Auswahllisten (z. B. `<!ATTLIST Name Wortform (John|Paul|George|...) #IMPLIED>`) stammen, sowohl als annotierende als auch als annotierte Daten betrachtet werden. So kann z. B. die Annotation des Satzes „John sang.“ folgendermaßen aussehen:

```
<s><np><Name Wortform='John'/></np><vp><vi>sang</vi></vp></s>
```

Diese Repräsentation scheint einerseits nur den annotierten Text ‚sang’ und sonst nur annotierende Daten zu enthalten, andererseits gehört jedoch das Wort ‚John’ zu den Primärdaten.

Aufnahme stoppt und das Deck schaltet sich aus“, einschließlich der verwendeten Leerzeichen, in allen Beispielen identisch.[34]

Den annotierten Daten bzw. den Primärdaten kommt bei der Verwendung des Ansatzes der separaten Annotierung eine besondere Bedeutung zu. Wenn dafür gesorgt wird, diese Daten konstant zu halten – wenn also in jeder der separaten Annotationen identische Primärdaten verwendet werden – können diese als Verknüpfung zwischen den Dokumenten der unterschiedlichen Annotationsebenen genutzt werden. Die Verknüpfung wird in diesem Fall implizit hergestellt, expliziert werden muss sie durch die Angabe einer Verknüpfungsprozedur. Die Herstellung der Verknüpfung wird im folgenden beschrieben. Hierfür kommt der Indizierung der Primärdaten eine zentrale Rolle zu.

PRIMÄRDATENINDIZIERUNG

Für jedes Zeichen z einer Annotation A_x, wobei x die Annotationsebene bezeichnet, gilt, dass z entweder Bestandteil der Menge der Annotationsdaten AD (z ϵ AD) oder Element der Menge der Primärdaten PD (z ϵ PD) ist. Jedem Zeichen z' innerhalb von A_x welches zu PD gehört (z' ϵ PD), wird ein Index i zugeordnet. Der Index i entspricht der Position von z' in der Liste der zu PD gehörenden Zeichen.

Nachfolgend soll diese Indizierung anhand zweier der oben aufgeführten Annotationsebenen exemplifiziert werden.

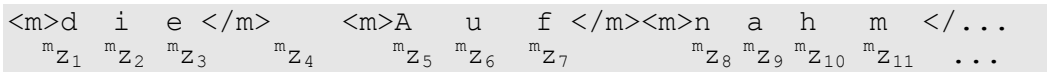
Beispielindizierung bzgl. der Annotation der Satzebene s:

```
<s><s>d   i   e      A  u  ...  t  </s>u   n   d       ...
     sz1  sz2 sz3 sz4 sz5 sz6 ... sz19    sz20 sz21 sz22 sz23  ...
```

In diesem Beispiel wird für die erste der oben wiedergegebenen separaten Annotationen verdeutlicht, wie die Daten indiziert werden. Der hochgestellte Index (hier: ^{s}) verweist auf die Annotationsebene, der tiefgestellte Index ($_1$, $_2$, $_3$, ...) referiert auf die Position des Zeichens innerhalb der Zeichenkette der zu den Primärdaten gehörenden Zeichen.

Beispiel für die Indexzuordnung auf der Morphemebene m:

[34] Die Zeilenumbrüche sind auf die visuelle Aufbereitung der Beispiele zurückzuführen. Sie werden nicht durch die Einfügung von Zeilenumbruchszeichen (CR/LF) ausgelöst.



Die Indizierung der Primärdaten für diese Ebene unterscheidet sich von der obigen ausschließlich durch einen anderen Annotationsebenenindex, also das voranstehende hochgestellte m für die Morphemebene ($^{m}z_i$), das an die Stelle der Kennzeichnung der Satzebene durch das hochgestellte s ($^{s}z_i$) tritt. Die Position der Zeichen in der Kette der Primärdaten ist unabhängig von der Gestalt der Annotationsdaten dieselbe.

Für jedes Zeichen der Primärdatenebene ist somit sichergestellt, dass es eindeutig und annotationsebenenübergreifend gekennzeichnet ist. Dadurch wird die implizite Verknüpfung der unterschiedlichen Annotationsebenen hergestellt. Bei Bedarf kann jederzeit auch eine explizite Verknüpfung aufgebaut und inkorporiert werden. Dies kann im einfachsten denkbaren Fall dadurch erfolgen, dass die Primärdaten, d. h. jedes Zeichen, in eine Umgebung (z. B. <c>) aufgenommen werden. Diese Umgebungen können dann mit dem ID-IDREF-Mechanismus miteinander verknüpft werden.[35]Um den Weg der separaten Annotation zur Repräsentation verschiedener logischer Ebenen beschreiten zu können, müssen jedoch mehrere Beschränkungen in der Datenhaltung vorgenommen werden. An erster Stelle ist hierbei die konsistente Datenhaltung und die Veränderbarkeit der Daten zu nennen.

Die Primärdaten müssen in allen der separaten Annotationen identisch sein. Dies bedeutet insbesondere, dass Veränderungen der Primärdaten in allen Annotationsebenen vorgenommen werden müssen. Dadurch wird vermieden, dass nach erfolgter Annotation die impliziten Referenzen zerstört werden können. Vergleichbar hierzu ist das ‚klassische' Problem, dass Referenzzielpunkte (die mit Attributen vom Typ ID gekennzeichneten Knoten) bei ihrer Tilgung besonders überprüft werden müssen, um die Zerstörung potentiell existierender Querverweise zu vermeiden. Da jedoch nicht nur bestimmte selten auftretende Arten von Daten betroffen sind, ist das Problem hier wesentlich virulenter.

[35] Eine solcher Weg dient nur der Illustration der Simplizität einer der vielen potentiellen Möglichkeiten zur expliziten Annotation der Verknüpfung. Bei der skizzierten Option handelt es sich um die einfachste, nicht die beste. Diese Diskussion ist ohnehin eher von theoretischem Belang, da durch das Vorhandensein der impliziten Verknüpfung keine Relevanz für diesen Schritt besteht.

Eine Reihe weiterer Punkte, die die konsistente Interpretation der Daten betreffen, ranken sich um das Thema Leer- bzw. Trennzeichenbehandlung:

- Es müssen Äquivalenzklassen der Zeichen vorgesehen werden. So ist es für eine Reihe von Dateninterpretationen irrelevant, ob es sich bei einem verwendeten Zeichen um einen Tabulatorenstop, ein Leerzeichen oder einen Zeilenumbruch handelt. Auch wenn dieses Problem bei einer starren Indizierung der Daten nicht notwendigerweise zu Verschiebungen der impliziten Verknüpfung führt, müssen diese Äquivalenzklassen aufgebaut werden, da erstens eine automatische Konsistenzprüfung der Primärdaten erfolgen können soll und zweitens die vorgestellte Indizierung eher einen modellhaften Charakter besitzt und die Verknüpfung praktisch eher über die vorhandene Datengleichheit hergestellt wird.
- Da Leerzeichen (bzw. deren Äquivalenzklasse) oft als einfache Trennzeichen verwendet werden, ist es in vielen Fällen irrelevant, ob eines dieser Zeichen auftritt oder ob mehrere hiervon direkt aufeinanderfolgend vorkommen. Dies zeigt sich an vielen Textstellen innerhalb HTML-annotierter Umgebungen, aber auch bei der Verarbeitung von Textdaten mit dem Satzsystem LaTeX (Lamport 1994). Allerdings gibt es auch Stellen in HTML und in LaTeX, in denen diese Zeichen nicht als Trennzeichen aufgefasst werden sollen. In diesen Umgebungen zählt jedes Auftreten dieser Zeichen. Der Standard XML reagierte auf den Bedarf der Unterscheidung beider Typen von Leerzeicheninterpretationen mit der Einführung eines allen Elementen zur Verfügung stehenden Attributs. Dieses Attribut (`xml:space`) kann zwei Werte annehmen, konkret den Wert `preserve` um jedes der Zeichen zu behalten und den Fall `default` durch den das Auftreten mehrerer Trennzeichen als das Vorkommen eines Leerzeichens interpretiert wird. Kommen separate Annotationen zur Anwendung, muss vermieden werden, dass in den unterschiedlichen Annotationen die potentiellen Trennzeichen unterschiedlich interpretiert werden. Da die Menge der Annotationsebenen unbegrenzt ist, also auch ständig erweitert werden kann, muss die Behandlung dieser Zeichen in einer strikt festgelegten Weise erfolgen.
- Das Auftreten der Trennzeichen wird in XML bzw. SGML-annotierten Texten an den Stellen vollständig ignoriert, an denen kein Dateninhalt

auftreten darf.[36] Ein solcher Fall wäre in einer Annotationsdefinition gegeben, bei der Texte ausschließlich in Wörter (<w>) und Interpunktionszeichen () eingeteilt sind und das Inhaltsmodell des Wurzelelementes folgendermaßen definiert wird: <!ELEMENT Text (w|i)+>. Eine Annotation des Textes „John sang!“ wäre: <Text><w>John</w ><w>sang</w>!<Text>, bei einer anderen Annotation, z. B. <s>John sang!</s>. Diese Annotationen und die Inhaltsmodelle stellen unter klassischen Bedingungen sinnvolle Modellierungen dar: das Worttrennzeichen in der ersten Annotation kann mühelos inferiert werden, in der zweiten Annotation ist es jedoch normaler Dateninhalt, der vorhanden sein muss. Im Fall der separaten Annotation der Annotationsebenen ist ein solches Vorgehen jedoch nicht akzeptabel, da sich die extrahierten Primärdaten unterscheiden, d. h. “Johnsang!“ gegenüber “John sang!“. Es muss gewährleistet sein, dass derartige Abweichungen nicht auftreten können. Eine mögliche Lösung besteht im Verzicht auf die Verwendung implizierter bzw. inferierter Daten. Das obige Leerzeichen kann z. B. als Interpunktion interpretiert werden. Hierdurch würde die Wortebenenannotation <Text><w>John</w> <w>sang</w>!</Text> lauten.

Werden all diese Punkte beachtet, bildet die multiple Annotation ein sehr mächtiges und adäquates Mittel zur Repräsentation der Informationen über multiple Annotationsebenen. In der nachfolgenden Diskussion sollen die Möglichkeiten der annotationsebenenübergreifenden Auszeichnung gegeneinander abgewogen und Wechselwirkungen der verschiedener Ansätze beleuchtet werden. Es werden des Weiteren Kombinationsmöglichkeiten und Mischformen der unterschiedlichen Wege diskutiert.

3.8 Diskussion

Lange Zeit ist davon ausgegangen worden, dass die den Texten zugrundeliegende Struktur hierarchischer Natur ist. Hierauf basieren die zum Zwecke der Textauszeichnung verwendeten Sprachen SGML etc. Mehr und mehr setzt sich jedoch die Auffassung durch, nicht alle Texte durchgängig mit derartigen, in dieser Hinsicht limitierten, Ansätzen beschreiben zu können. Die Erkenntnis

[36] Vgl. auch Abschnitt „Mixed-Content“ (ab S. 33)

dieser gravierenden Defizite einer strikt hierarchischen Textstrukturierung führte zur Betrachtung einer Reihe von technischen Lösungsvorschlägen.

Vom SGML-Standard selbst ist die Verwendung einer erweiterten Syntaxmöglichkeit vorgesehen, die es erlaubt, Texte gleichzeitig nach verschiedenen, möglicherweise gegensätzlichen Dokumentgrammatiken auszuzeichnen. Dies wird durch die Änderung des Wertes der Standardeinstellung des Attributes `CONCUR` in der SGML Deklaration erwirkt. Die Verwendung dieser Möglichkeit bringt jedoch neben dem Vorteil, eine direkte und genuin SGML-basierte Lösung für derartige Fälle zu besitzen, so viele Nachteile mit sich, dass selbst der Hauptautor des SGML-Standards, Charles Goldfarb, von ihrer Verwendung abrät.

Andere Möglichkeiten der Repräsentation von Texten, deren Struktur nicht als geordnete Hierarchie von Inhaltsobjekten beschrieben werden kann, liegen in der Verwendung von Meilensteinen (vgl. Abschnitt 3.3) und in der Nutzung von Verknüpfungen (vgl. 3.2), insbesondere wenn eine Verknüpfung mit einer Zeitachse erfolgt (Abschnitt 3.2.2). Trotz der Unterschiedlichkeit dieser Ansätze besitzen sie eine Gemeinsamkeit, die von grundsätzlicher Natur ist. Da SGML und insbesondere XML nur eine Hierarchie repräsentieren können, müssen die weiteren Hierarchien anderweitig modelliert werden. Diese anderweitige Modellierung besteht in der Sequentialisierung der (zusätzlichen) hierarchischen Strukturen.

Verdeutlicht werden soll dies anhand zweier Ausschnitte aus den obigen Beispielannotationen, bei denen jedoch ein Wurzelelement eingefügt wurde, um dieses für wohlgeformte XML-Dokumente notwendige Kriterium zu erfüllen. Des Weiteren wurde die Daten- und Elementebene durch die Annotation der vorhandenen Leerzeichen (mit `<sp>`) getrennt, um das Auftreten von problematischem ‚mixed-content' zu verhindern.

Die Silbenstruktur und die morphologische Struktur der Wörter des Satzes „die Aufnahme stoppt" unterscheidet sich beim Wort ‚Aufnahme', dergestalt, dass eine simple Annotation der beiden Einheiten innerhalb eines XML-Dokumentes nicht zugelassen wäre.

Naive Annotation:

```
<root><syll><m>die</m></syll><SP> </SP><syll><m>Auf</m></s
yll><syll><m>nah</syll><syll>m</m><m>e</m></syll><SP> </SP
><syll><m>stopp</m><m>t</m></syll></root>
```

Es ist zu sehen, dass diese – möglicherweise naheliegende – mehrere Hierarchien markierende Repräsentation gegen das Wohlgeformtheitskriterium „Element Type Match“[37] verstößt. Erlaubt wäre dies nur bei der Verwendung der oben mehrfach besprochenen SGML-Möglichkeit CONCUR. Ihr Einsatz würde analog zu der nicht-wohlgeformten XML-Instanz aufgebaut sein, was hier eine Wiedergabe überflüssig werden lässt.

Die häufigste Art, widersprüchliche Hierarchisierungen zu beschreiben, erfolgt mit Hilfe der oben unter dem Begriff der Sequentialisierung zusammengefassten Methoden. Die erste der beiden Möglichkeiten, die faktisch zu einer Sequentialisierung der Hierarchien führen, bildet in dem dargestellten Beispiel die Markierungen der Silbengrenzen durch leere Elemente, die sogenannten Meilensteinen. Dabei soll hier der Beginn einer Silbe durch `<sb/>`, das Ende einer Silbe mit `<se/>` markiert werden.

```
<root><m><sb/>die<se/></m><SP> </SP><m><sb/
>Auf<se/></m><m><sb/>nah<se/><sb/>m</m><m>e<se/
></m><SP> </SP><m><sb/>stopp</m><m>t</m><se/></root>
```

Eine derartige Textrepräsentation ermöglicht den Zugriff auf die mit `<m>` markierten Morphe in einer deklarativen Weise. Diese Deklarativität soll anhand einer kontrastiven Darstellung der Zugriffsmöglichkeiten auf die Morphe und auf die Silben erfolgen. Hierfür ist es sinnvoll, die baumartige Repräsentation dieser Datei zu betrachten. Die Darstellung erfolgt im ESIS-Format (vgl. Abschnitt 2.1.3.3).

[37] Die XML-Spezifikation definiert eine Reihe dieser Kriterien zur Wohlgeformtheit und Gültigkeit von XML-Elementen. Das „Well-formedness constraint: Element Type Match“ drückt genau die der OHCO-These zugrundeliegende Beschränkung aus. Sie lautet: „The name in an element's end-tag must match the element type in the start-tag.“ (Bray et al. 2000)

```
(root (m (sb)sb -die (se)se )m
      (SP - )SP
      (m (sb)sb -Auf (se)se )m
      (m (sb)sb -nah (se)se (sb)sb -m)m
      (m -e (se)se )m
      (SP - )SP
      (m (sb)sb -stopp)m
      (m -t )m (se)se
)root
```

Die Klammerung, die an die Syntax der Programmiersprache LISP erinnert, spiegelt die Baumstruktur wider.

Anhand der Darstellung soll jetzt verdeutlicht werden, dass auf bestimmte Ausschnitte des Dokumentes relativ einfach zugegriffen werden kann, wohingegen andere Ausschnitte nur mittels komplexerer Zugriffsoperationen extrahiert werden können. Die Repräsentation verwendet Klammerpaare, die Informationen umschließen. Es ist sehr einfach, auf den *vollständigen* Inhalt eines oder mehrerer dieser Klammerpaare zuzugreifen. Dieses Zugreifen kann in einer aufeinanderfolgenden Weise erfolgen, so dass Kombinationen dieser Zugriffe möglich werden. Es muss bei diesen Kombinationen nur gewahrt bleiben, dass immer die gesamten Klammerinhalte verarbeitet werden.

Sollen aus der obigen Repräsentation des (Teil-)Satzes „die Aufnahme stoppt“ alle Morphe herausgefiltert werden, kann in einem ersten Schritt angegeben werden, dass sämtliche als <m> annotierte Umgebungen extrahiert werden sollen, d. h. das erste, dritte, vierte, fünfte, siebte und achte direkte Nachfolgeelement wird herausgegriffen:

```
(m (sb)sb -die (se)se )m
(m (sb)sb -Auf (se)se )m
(m (sb)sb -nah (se)se (sb)sb -m)m
(m -e (se)se )m
(m (sb)sb -stopp)m
(m -t )m
```

In einem weiteren Schritt werden alle leeren Unterelemente, wiederum vollständige Klammerpaare, herausgegriffen und getilgt. Das Ergebnis ist dann:

```
(m die)m
(m Auf)m
(m nahm)m
(m e)m
(m stopp)m
(m t )m
```

Im Gegensatz zu einer derartigen Verarbeitung ist es für die Extraktion der mit Meilensteinen annotierten Silben nicht möglich, diesen Weg zu wählen. Exemplarisch soll gezeigt werden, welche Probleme entstehen, versuchte man die oben mit Meilensteinen annotierte Silbenstruktur des Wortes ‚Aufnahme' herauszufiltern.

Die Silben des Wortes ‚Aufnahme' sind in den Umgebungen dreier mit <m> ausgezeichneten Umgebungen enthalten. Sie befinden sich jeweils zwischen dem Ende des (leeren) Elementes <sb> und dem Beginn des (leeren) Elementes <se>.

```
(m (sb)sb -Auf (se)se )m
(m (sb)sb -nah (se)se (sb)sb -m)m
(m -e (se)se )m
```

Es ist zu erkennen, dass es nicht möglich ist, ein Klammerpaar zu bestimmen, dessen Inhalt denen der Silben entspricht. Zwar zeigt die erste Zeile die Zeichenkette ‚Auf', die sich zwischen einer sb- und einer se-Markierung befindet, und die ihrerseits in ein Klammerpaar eingeschlossen ist. Allerdings enthält dieses Klammerpaar das Morph ‚auf', welches zufälligerweise mit der die Silbe ‚auf' denotierenden Zeichenkette übereinstimmt. Deutlicher wird das Problem anhand der beiden weiteren Silben ‚nah' und ‚me'. Die erste ist nur ein Teil einer in Klammern eingeschlossenen Umgebung. Da jedoch nur auf den vollständigen Inhalt eines Klammerpaares zugegriffen werden kann, ist die Silbe ‚nah' nicht deklarativ extrahierbar. Die Filterung der Silbe ‚me' ist noch problematischer, da sie nur durch die Konkatenation des unvollständigen Inhalts zweier <m>-Umgebungen ermittelt werden kann. Es bleibt zu fragen, wie es möglich ist, auf die vollständigen Silben zuzugreifen, die – wenn auch mit leeren Elementen – eindeutig markiert wurden:

Die interne Repräsentation des annotierten Dokumentes, im obigen Beispiel im ESIS-Format, als eine von verschiedenen Möglichkeiten[38], ist das Resultat

[38] SGML- und XML-Dokumente werden für die Zwecke der effektiveren maschinellen Verarbeitung oder der Einfügung einer Abstraktionsebene in baumähnliche Repräsen-

einer Verarbeitung. Das Dokument wird als Sequenz von Zeichen eingelesen, überprüft und mit Hilfe eines Parsers in eine derartige Form überführt. Es war anhand des Morphbeispiels zu sehen, dass bestimmte Informationen in deklarativer Weise aus dem Resultat dieser Verarbeitung extrahierbar sind. Die mit Meilensteinen markierten Einheiten gehören nicht zu diesem Typ von Einheiten. Wenn versucht wird, aus dieser Repräsentation die Silben herauszugreifen, muss unabhängig von der bereits durch den Parser aufgebauten Baumrepräsentation nach weiteren Sequenzen gesucht werden. Alle sich zwischen `(sb)sb` und `(se)se` befindenden Zeichen müssen für diesen Zweck gesucht werden. Folgendes Ergebnis kann aus diesem Verarbeitungsschritt resultieren:

```
(sb)sb -Auf (se)se
(sb)sb -nah (se)se
(sb)sb -m)m (m -e (se)se
```

In einem weiteren Schritt muss das Ende und der Beginn der ursprünglich mit `<m>` gekennzeichneten Umgebungen – ein nicht zueinanderpassendes Klammerpaar – getilgt werden, um das im folgenden angegebene gewünschte Ergebnis zu erhalten:

```
(sb)sb Auf (se)se
(sb)sb nah (se)se
(sb)sb me (se)se
```

Die beiden zuletzt genannten Arbeitsschritte benötigen jedoch ihrerseits wiederum den Einsatz einer Verarbeitungsmaschinerie, z. B. eines speziellen Parsers, der als Eingabe das Resultat eines allgemeinen SGML-Parsers erhält.

Diese längere Ausführung exemplifiziert das gravierende Problem bei der Nutzung von Dokumenten, die Meilensteine als Markierung verwenden: ein Problem, welches sich qualitativ von den am Ende des Abschnitt 3.3 genannten Problemen unterscheidet. SGML und XML erlauben die Modellierung von Dokumenten mit Hilfe von frei definierbaren Dokumentgrammatiken. Diese Dokumente können die abstrakte, innerhalb von Modellen (z. B. den DTDs)

---

tationen überführt. Neben dem ESIS sind verschiedene weitere dieser Formate entwickelt worden. Zu abstrakten Formaten gehören die ‚Graph Representation of Property Values' (Grove, definiert in einem Anhang A.4 des HyTime-Standards ISO/IEC 10744) und der XPath Tree (Clark und DeRose 1999), zu den spezielleren zählt das Document Object Model (DOM, vgl. World Wide Web Consortium).

beschriebenen Informationen in einer deklarativen Weise beinhalten. Unabhängig davon ist es möglich, weitere Informationen in einer resolvierbaren Form in den Dokumenten aufzunehmen. Die Nutzung dieser Möglichkeit widerspricht jedoch den Grundprinzipien der Informationsmodellierung. Folgerichtig finden derartige Konstrukte in sauber modellierten DTDs nur sehr selten Verwendung, obwohl es problemlos möglich wäre.[39] Auf sie wird nur zurückgegriffen, wenn eines der Grundprinzipien des SGML-Standard, die OHCO-These, einer deklarativen Lösung im Wege steht.

Auch bei der im Abschnitt 3.2.2 thematisierten Verwendung eines Zeitstrahls (bzw. der oben beschriebenen äquivalenten Primärdatenebene) als Verknüpfungsebene erfolgt die Annotation bestimmter Einheiten nicht über Markierungen von Umgebungen, sondern mittels Verweisen auf punktuelle Informationen und führt folglich ebenfalls zu einer Sequentialisierung von bestimmten Hierarchien – mit all den gerade beschriebenen Problemen.

Bei der Verwendung der im Abschnitt 3.2.1 angesprochenen Verweise auf eine primäre Annotation werden Verknüpfungen als Mittel der Inklusion einer separaten Annotationsebene gebraucht. Dies bildet eine Alternative zur Annotation multipler Hierarchien mit Meilensteinelementen, die den Vorteil mit sich bringt, dass durch diesen Ansatz die Deklarativität der Annotation gewahrt bleiben kann. Ein in einer MATE-Veröffentlichung (Mengel et al. 2000:223f.) vorgestelltes Beispiel annotiert den (unterbrochenen) Satz „The thing - no you cannot open the door now - suddenly spoke.“:

```
...  <w id="w_001">The</w>
     <w id="w_002">thing</w>
     <w id="w_003">no</w>
     <w id="w_004">you</w>
     <w id="w_005">cannot</w>
     <w id="w_006">open</w>
     <w id="w_007">the</w>
     <w id="w_008">door</w>
     <w id="w_009">now</w>
     <w id="w_010">suddenly</w>
     <w id="w_011">spoke</w>  ...
```

Wird nun der Ausschnitt „The thing suddenly spoke“ als Satz annotiert, wird in einer separaten Annotation auf die zu dem Satz gehörenden annotierten und

[39] Beispielsweise wäre es sehr einfach möglich gewesen, innerhalb von HTML den Beginn und das Ende von hervorgehobenen Textteilen mit `<begin-em>` bzw. `<end-em>` zu kennzeichnen.

über ihre ID referenzierbaren Einheiten – hier die mit <w> markierten Wörter – verwiesen. Eine der möglichen Annotationen des Satzes ist:

```
... <s id="s_001">
      <wlink id="wlink_001"
             href="normword2.xml#id(w_001)"/>
      <wlink id="wlink_002"
             href="normword2.xml#id(w_002)"/>
      <wlink id="wlink_003"
             href="normword2.xml#id(w_010)"/>
      <wlink id="wlink_004"
             href="normword2.xml#id(w_011)"/>
    </s>
```

Die Deklarativität der Annotation kann gewahrt bleiben, da die Satzauszeichnung so aufgebaut ist, dass hierbei verschiedene vollständige Umgebungen aus der Primärdatenannotation extrahiert und zu einem neuen Ganzen zusammengefügt werden. Nichtsdestotrotz kann nicht übersehen werden, dass die Interpretation des Satzes auch hier in einem zweiten Schritt erfolgt. Die naheliegende Modellierung, dass ein Satz aus Wörtern besteht, wurde hier aus technischen Gründen um die Möglichkeit erweitert, dass ein Satz aus Referenzen auf Wörter bestehen kann. Wenn diese Möglichkeit genutzt wird, müssen diese Referenzen erst aufgelöst werden, d. h. die referierten Einheiten müssen in einem ersten Schritt gewonnen werden. Trotz dieses Punktes bildet der von MATE gewählte Ansatz der Annotation multipler Hierarchien eine bessere Modellierungsgrundlage. Der gravierendste Nachteil dieses Ansatzes bleibt – aus der Perspektive der Informationsmodellierung[40] betrachtet – der oben erwähnte Punkt der Granularität der Annotationsbasis. Je gröber die Basisannotation ist, desto wahrscheinlicher ist es, dass die Basisannotation erweitert werden muss, da anfangs nicht beachtete Annotationsebenen eine detailliertere Segmentierung der Basis erforderlich werden lassen.

Die Deklarativität bleibt auch bei der „Fragmentierung von Elementen“, die in 3.4 beschrieben wird, bestehen. Die TEI-Richtlinien weisen jedoch auch auf einen wichtigen Nachteil hin, der allerdings nicht nur diesem Ansatz inhärent ist: Es findet im Vorhinein eine Auswahl mit dem Ziel statt, dass bestimmt

[40] Eine weitere Problemebene ist die Unterstützung durch existierende generische SGML- bzw. XML-Software. Das Projekt MATE entwickelte für den Annotationsrahmen eine spezielle Annotationssoftware, die sogenannte MATE-workbench.

wird, welche Elemente fragmentiert werden und welche nicht. Dieser Umstand privilegiert implizit bestimmte Umgebungen.

Im Abschnitt „Implizite Verknüpfung der Auszeichnungsebenen“ (Abschnitt 3.5) wurden zwei verschiedene Interpretationen von Annotationen vorgestellt, zum einen die implizite Verbindung von nicht miteinander in Beziehung gesetzten Annotationen und zum anderen die theorieinhärenten impliziten Bezüge. Der erste Weg lässt sich durch die in Absatz 3.7 vorgestellten separaten Annotationen formal greifbarer explizieren. Die zweitgenannte Interpretation legt eine Modellierung mit architektonischen Formen nahe. Um diese Wege beschreiten zu können, müssen allerdings Techniken verwendet werden, die noch nicht zu den Standardmethoden der Texttechnologie gehören.

Das zu lösende Grundproblem bei der Nutzung der impliziten Verknüpfungen besteht darin, dass sie nur dann theoretische und insbesondere praktische Bedeutung besitzen können, wenn sie in explizite Verknüpfungen verwandelt werden. Für derartige Explizierungen bieten sich wiederum die Möglichkeiten der Angabe architektonischer Beziehungen zwischen den Dokumentgrammatiken an, wobei allerdings die oben beschriebenen Limitierungen der architektonischen Verarbeitung greifen.

4 Dokumente, Dokumentgrammatik und Unifikation

Die Linguistik bedient sich seit mehr als zwanzig Jahren der sogenannten Unifikationsgrammatiken. In dieser Zeit ist eine Vielzahl von Grammatiken entwickelt worden, die sich als unifikationsbasiert klassifizieren lassen. Zu ihnen gehören die phrasenstrukturbasierten Lexical Functional Grammar (LFG, Bresnan 1982, 2001), die Head-driven Phrase Structure Grammar (Pollard & Sag 1996, St. Müller 1999), kategorialgrammatisch fundierte Syntaxformalismen (z. B. Uszkoreit, 1986) und Dependenzgrammatiken (Hellwig, 1986; Witt, 1996). Wie schon aus der Aufzählung sichtbar wird, sind die Unifikationsgrammatiken keineswegs einer einheitlichen linguistischen Schule entsprungen – was sie verbindet ist ausschließlich der Formalismus.

Namengebend für diese Gruppe der Grammatiken ist die Unifikation. Diese Operation erlaubt es, partielle Informationen zusammenzuführen. Dies hat zur Folge, dass komplexere Informationen aus Teilinformationen zusammengesetzt werden können oder dass Inkompatibilität partieller Informationen festgestellt wird.

In diesem Kapitel soll die Funktionsweise dieser Formalismen kurz vorgestellt (Absatz 4.1) werden. Da diese Theorien, abgesehen von ihrer Verwendung zum Zwecke der maschinellen Sprachverarbeitung, auch eine große Bedeutung für die deklarative Repräsentation sprachlichen Wissens besitzen, sollen Vorschläge, die Basisformalismen mittels SGML bzw. XML zu beschreiben, diskutiert werden (Absatz 4.2). Diese Diskussion hat zum Ziel zu explorieren, inwieweit die direkte Inkorporation eines derartig formalisierten grammatischen Wissens mit Hilfe XML-basierten Annotationen sinnvoll sein kann.

Das Hauptanliegen der folgenden Darlegung ist allerdings relativ unabhängig von der Anwendungsdomäne der maschinellen Sprachverarbeitung zu sehen. Die vielfache Anwendung der Unifikationsoperation in der Linguistik ist darauf zurückzuführen, dass mit ihr eine Methode zur Verfügung steht, die es ermöglicht, Informationen aus verschiedenen Ressourcen zusammenzuführen. Es können damit separate Teilinformationen zu informationshaltigeren Einheiten zusammengesetzt werden. Es soll nachfolgend gezeigt werden, dass dies auch für die mit SGML bzw. XML modellierten Textdaten eine Perspektive sein kann, die erlaubt, verteilt vorliegende Informationen zu vereinen (Absatz 4.3).

4.1 Unifikationsbasierte Grammatiken

4.1.1 Modelle und Merkmalsstrukturen

Die Unifikation verwendet als Eingabe deklarativ repräsentierte Daten und liefert, im Falle einer erfolgreichen Anwendung, ein deklarativ repräsentiertes Ergebnisdatum. Die zu verarbeitenden Daten werden in einem Formalismus ausgedrückt, der auf gerichteten Graphen basiert. Die neueren theoretischen Spezifikationen geben sich jedoch nicht mehr mit den anfangs etwas vagen Spezifikationen der Datenformate zufrieden. Vielmehr wird ein theoretisches Fundament aufgebaut, welches es erlaubt, genauer über die Natur der zu unifizierenden Entitäten zu sprechen.

Die Basis derartiger linguistischer Ansätze bilden Modellierungen sprachlicher Einheiten. Diese Modelle werden mittels gerichteter Graphen formal notiert, die Merkmalsstrukturen (engl.: feature structure) genannt werden. Eine Merkmalsstruktur enthält das gesamte, d. h. vollständige, Modell des betreffenden Weltausschnitts, wobei der Terminus ‚gesamt' bzw. ‚total' abgeschwächt wird:

> To get a notion of "total" information, we develop a notion of maximal feature structures that represent maximal consistent amounts of information. But we do not take these objects to represent total information about an object in the real world in absolute terms, since it is unlikely that a cognitive agent would ever have total information about an object other than relative to some fixed stock of concepts ...
>
> Carpenter (1992:51)

Zweifellos steht allerdings fest, dass die Modelle das Wissen vereinen, welches maximal durch die Sprachverarbeitungskomponente ermittelt werden kann. Die Modelle sind jedoch nicht selbst Teil der Grammatik, sie bilden vielmehr den Beschreibungsgegenstand.

Beim Prozess der maschinellen Sprachverarbeitung wird versucht, aus verschiedenen Quellen Informationen zusammenzutragen und sich somit dem maximal spezifizierten Modell anzunähern. Diese Annäherung geschieht in der Form, dass ebendiese Merkmalsstrukturen beschrieben werden.

4.1.2 Beschreibungen von Merkmalsstrukturen

Nun ist es so, dass Merkmalsstrukturen unabhängig von ihrer Fähigkeit, als ideales Modell fungieren zu können, auch den besten Formalismus bilden, um

Merkmalsstrukturen zu beschreiben. Dies führt häufig zu Verwirrungen (vgl. auch Zitat in der Fußnote 42 auf Seite 89), da oft nicht strikt zwischen der Ebene der Merkmalsstruktur und der Ebene ihrer Beschreibung unterschieden wird. Die häufig verwandte Lösung besteht in der Benutzung unterschiedlicher Notationen für Merkmalsstrukturen und für die Beschreibungen von Merkmalsstrukturen. Die Merkmalsstrukturen selbst werden in einer Graphenschreibweise notiert, die Beschreibungen in einer Matrizenschreibweise. Der Basisformalismus der unifikationsbasierten Grammatiken, die zu manipulierenden Objekte, bilden selbstverständlich die Beschreibungen von Merkmalsstrukturen, da die Merkmalsstrukturen selbst als totale Objekte gar nicht manipuliert werden können.

4.1.2.1 Aufbau von Merkmalsstrukturbeschreibungen

Eine Beschreibung einer Merkmalsstruktur, die Attribut-Wert-Matrix (AWM), spezifiziert endlich viele Merkmale. Jedes Merkmal besitzt einen Wert. Alle spezifizierten Merkmale besitzen Gültigkeit, d.h. es werden nur Merkmale beschrieben, die in der zu beschreibenden Merkmalsstruktur vorhanden sind. Daraus ergibt sich, das die Merkmale implizit mit dem logischen UND verknüpft sind. Eine AWM die keine Merkmale spezifiziert, heißt leere AWM. Sie wird so interpretiert, dass sie alle Merkmalsstrukturen, d. h. alle (modellierten) Dinge in der Welt beschreibt.

Alle nichtleeren Attribut-Wert-Matrizen spezifizieren n (n > 1) Merkmal(e). Die Reihenfolge der aufgeführten Merkmale ist unerheblich, da alle spezifizierten Eigenschaften gelten. Die Werte der Merkmale können atomar oder komplex sein. Beispiele für atomare Werte sind +, *singular* und 3. Im Gegensatz zu den älteren Verwendungen von Attribut-Wert-Spezifikationen[41] können AWMs auch Merkmale besitzen, deren Werte nicht atomar sind. Diese komplexen Merkmalsstrukturen zeichnen sich dadurch aus, dass der Wert eines Merkmals auch eine AWM ist.

Eine fundamentale Eigenschaft von Attribut-Wert-Matrizen ist, dass verschiedene Attribute auf denselben Wert zugreifen können, d.h. ein Wert kann Spezifikation mehrerer Attribute sein. Derartige Strukturen werden als co-indiziert,

[41] Simons und Langendoen (1995) schreiben die erste Verwendung von Merkmalsstrukturen Roman Jakobson zu, der 1949 Merkmale zur Klassifizierung von Phonemen verwendete, die jeweils binäre Werte annehmen konnten, z. B. stimmhaft +/-. Chomsky dehnte ihre Verwendung auf die Ebene der Syntax und die der Morphologie aus.

von Haugeneder und Trost (1993) als koreferent bzw. pfadäquivalent und im Englischen u. a. als ‘reentrant’ bezeichnet. Wie die Vielzahl verschiedener Benennungen dieser Eigenschaft möglicherweise schon vermuten lässt, ist diese Eigenschaft oft thematisiert worden. In der Tat sind es solche Gleichsetzungen, die in allen unifikationsbasierten Grammatiken von einer besonderen Relevanz sind, da durch sie die in allen Sprachen vorkommende Beziehungen zwischen den Worten eines Satzes, zwischen Lexikoneinträgen und Phrasen, zwischen den Wörtern und ihrer Bedeutung oder der Komposition der Satzbedeutung aus der Semantik der Satzteile spezifiziert werden können.

4.1.2.2 Eigenschaften und die Operation ‚Unifikation’

Die zentrale und namensgebende Operation bei unifikationsbasierten Formalismen heißt *Unifikation*. Um sie darzustellen, sollen jedoch zunächst Eigenschaften von Merkmalsstrukturbeschreibungen betrachtet werden, die für diese Operation von Bedeutung sind.

Eine merkmalsstrukturenbeschreibende AWM kann mehr oder weniger konkret sein. Beispielsweise ist die leere AWM sehr unpräzise – mit ihr können alle Merkmalsstrukturen beschrieben werden – allerdings nur partiell. Unterschiedlich spezifische Beschreibungen von Merkmalsstrukturen können zueinander in einer Beziehung, der Subsumption, stehen.

SUBSUMPTION VON ATTRIBUT-WERT-MATRIZEN

Eine AWM D subsumiert eine AWM D’ (D $\sqsubseteq$ D’) genau dann, wenn alle Informationen, die in D vorhanden sind auch in D’ enthalten sind. Dies bedeutet, dass alle von D’ beschriebenen Attribut-Wert-Strukturen auch von D beschrieben werden.

Aus einer mengentheoretischen Perspektive heraus kann dies dergestalt paraphrasiert werden, dass die subsumierte AWM eine Teilmenge der von der subsumierenden AWM beschriebenen Modelle beschreibt. Es ist bei dieser Sichtweise leicht zu erkennen, dass die leere AWM jede Merkmalsstruktur subsumiert, da durch sie jedes Modell, also alle Elemente der Menge aller Modelle, beschrieben wird. Die Qualität der Beschreibung bleibt hierbei unberücksichtigt.

Zwei Attribut-Wert-Matrizen müssen sich jedoch nicht in einer Subsumptionsbeziehung zueinander befinden. Eine nicht existierende Subsumptionsbezie-

hung zweier Merkmalsstrukturbeschreibungen kann auf zwei verschiedene Ursachen zurückgeführt werden:

- zwei Attribut-Wert-Matrizen beinhalten unterschiedliche, aber kompatible Informationen oder
- zwei Attribut-Wert-Matrizen beinhalten inkompatible, d.h. widersprüchliche Informationen

Auch diese Formen der Beziehungen lassen sich mengentheoretisch genauer ausführen. Zueinander inkompatible Informationen beinhaltende A-W-Matrizen beschreiben zueinander disjunkte Teilmengen, z. B. die Menge der Tiere mit 6 Beinen gegenüber der Menge der Tiere mit 4 Beinen oder (in Grammatiktheorien häufiger zu finden) die Menge der maskulinen Nomen gegenüber der der femininen. Kompatibel zueinander ist hingegen die Beschreibung aller Elemente aus der Menge der Nomen, deren Genus feminin ist (z. B. Frau, Uhr) gegenüber der Beschreibung aller Nomen, die weibliche Lebewesen bezeichnen (z. B. Frau, Mädchen).

Bei einer genaueren Betrachtung kompatibler und inkompatibler AWMs kann festgestellt werden, dass die in kompatiblen Merkmalsstrukturen enthaltenen unterschiedlichen Informationen durch eine AWM beschrieben werden können. Spezifiziert z. B. eine AWM im Plural stehende Wörter und eine andere AWM die maskulinen Nomen, dann kann eine Merkmalsstrukturbeschreibung aufgebaut werden, in denen diese drei Eigenschaften widerspruchsfrei spezifiziert werden.

Eine Attribut-Wert-Matrix, die exakt und ausschließlich die Informationen zweier kompatibler Merkmalsstrukturen enthält, heißt allgemeinster Unifikator dieser beiden Matrizen. Der allgemeinste Unifikator ist das Ergebnis der zentralen Operation der Unifikationsgrammatiken, der Unifikation.

DIE OPERATION UNIFIKATION

Das Ergebnis der Unifikation zweier Attribut-Wert-Matrizen D' und D'' ist die allgemeinste Attribut-Wert-Matrix D, die sowohl von D' als auch von D'' subsumiert wird. Wenn keine solche Struktur existiert, ist die Unifikation nicht definiert. Die Unifikation wird mittels des Symbols ⊔ dargestellt. Das Ergebnis von D' ⊔ D" – beschreibt alle Merkmalsstrukturen, die sowohl von D' als auch von D'' beschrieben werden.

Mit Hilfe dieser Operation ist es somit möglich, die Kompatibilität verschiedener AWMs zueinander zu überprüfen, mehrere kompatible Beschreibungen

linguistischer Objekte zu kombinieren und in einer adäquaten Form zu repräsentieren. Für die Unifikation von Merkmalsstrukturbeschreibungen gelten die folgenden Gesetze:

- Kommutativitätsgesetz: AWM-I ⊔ AWM-II = AWM-II ⊔ AWM-I
- Assoziativitätsgesetz: (AWM-I ⊔ (AWM-II ⊔ AWM-II)) = ((AWM-I ⊔ AWM-II) ⊔ AWM-III)
- Verschmelzungsgesetz bzw. Idempotenzgesetz: AWM-I ⊔ AWM-I = AWM-I

Diese Gesetze haben zur Folge, dass die Reihenfolge der Abarbeitung der Verarbeitungsschritte bei der maschinellen Verarbeitung irrelevant ist. Die Deklarativität der Verarbeitung kann durch die Gültigkeit dieser Gesetze sichergestellt werden.

Dass diese Gesetze gelten, ist insbesondere auf das Design der Attribut-Wert-Strukturen und ihrer Beschreibungen zurückzuführen: Merkmalsstrukturen spezifizieren ausschließlich das Vorhandensein von Eigenschaften. Dadurch ergibt es sich von selbst, dass die Reihenfolge der Merkmalsaufführungen unbedeutend ist.

4.1.3 Beschreibungen getypter Merkmalsstrukturen

Getypte Attribut-Wert-Strukturen werden durch getypte Attribut-Wert-Matrizen beschrieben. Getypte AWMs unterscheiden sich durch die Angabe eines Typs von einfachen AWMs. Ein solcher Typ denotiert einen Typen in einem Typensystem.

Ein Typ einer Attribut-Wert-Struktur bestimmt, welche Merkmale diese Attribut-Wert-Struktur besitzt. Es wird davon gesprochen, dass Typen Angemessenheitsbedingungen aufstellen. Wenn eine Attribut-Wert-Struktur Merkmale besitzt, die von dem zugehörigen Typ nicht bestimmt sind, ist die Attribut-Wert-Struktur nicht korrekt getypt, oder – positiv formuliert – wenn eine Attribut-Wert-Struktur ausschließlich Merkmale besitzt, die durch ihren Typ legitimiert sind, ist diese Attribut-Wert-Struktur richtig getypt (engl.: *well-typed*). Wenn alle durch ihren Typ erlaubten Merkmale einer Attribut-Wert-Struktur spezifiziert wurden, wird davon gesprochen, dass die Attribut-Wert-Strukturen vollständig richtig getypt (engl.: *totally well-typed*) sind.

4.1.3.1 Das Typensystem

Das Typensystem besteht aus einer endlichen Menge von Typen. Die Typen besitzen einen unterschiedlichen Grad der Allgemeinheit. Dies bedeutet, dass der Informationsgehalt der Typen unterschiedlich ist. Manche Typen beschreiben Entitäten, die auch von anderen Typen beschrieben werden. Dies kann zwei Ursachen haben. Zum einen ist es möglich, dass zwei (oder mehr) Typen denselben Informationsgehalt besitzen. In diesem Fall sind die Typen nicht wirklich unterschiedlich, es handelt sich vielmehr um alphabetische Varianten desselben Typs. Wenn einer von zwei Typen jedoch einen höheren Informationsgehalt als der andere besitzt, wird davon gesprochen, dass der mit dem höheren Informationsgehalt spezifischer ist als der Typ mit dem geringeren Informationsgehalt. Das Typensystem besitzt eine Ordnung. Die Ordnung richtet sich nach der Spezifik der enthaltenen Typen.

Es ist hervorzuheben, dass die durch das Typensystem erreichte Ordnung bewirkt, dass allgemeinere Typen ihren Informationsgehalt an spezifischere Typen vererben.

Es sei die Konvention eingeführt, dass Variablen, deren Werte Typen sind, mit kleinen griechischen Buchstaben (α, β, γ, δ, …) bezeichnet werden. Mengen von Typen werden mit griechischen Großbuchstaben (Γ, Δ, Θ, …) bezeichnet. Ein Typ τ ist spezifischer als ein Typ σ, wenn τ Informationen von σ erbt. In einem solchen Fall wird gesagt, dass σ τ subsumiert:

SUBSUMPTION VON TYPEN

σ subsumiert τ ($\sigma \sqsubseteq \tau$), wobei σ und τ $\in$ Type, genau dann wenn jedes Objekt, welches vom Typ τ ist, auch vom Typ σ ist.

Wenn ein Typ σ einen Typ τ subsumiert, wird davon gesprochen, dass σ ein Supertyp des Typs τ ist und τ ist ein Subtyp von σ. Ein Subtyp des Typs σ ist spezifischer und ein Supertyp allgemeiner als σ.

Die in den neueren Grammatiktheorien verwendeten Typensysteme müssen den folgenden Anforderungen genügen:

- Das Typensystem besteht aus einer Menge von Typen.
- Die Ordnung des Typensystems wird durch die Subsumptionsrelation ausgedrückt.
- Zwei Typen des Typensystems, die gemeinsame Subtypen besitzen, besitzen genau einen gemeinsamen allgemeinsten Subtyp.

Da das Typensystem die Kompatibilität von Informationseinheiten in ihre Hierarchie integriert, muss bei seiner Definition bereits beachtet werden, welche Merkmale die Attribut-Wert-Strukturen der entsprechenden Typen besitzen. Der Typ einer getypten Merkmalsstruktur legitimiert durch die Vererbung innerhalb der Typenhierarchie mindestens all die Merkmale, die ihr Supertyp auch erlaubt.

4.1.3.2 Getypte Merkmalsstrukturbeschreibungen

Getypte Merkmalsstrukturbeschreibungen erweitern die Attribut-Wert-Matrizen aus Abschnitt 4.1.2.1 in der Form, dass sie Merkmalsstrukturen beschreiben können, die mit Typen versehen sind. Attribut-Wert-Matrizen, die getypte Attribut-Wert-Strukturen beschreiben können, heißen getypte Attribut-Wert-Matrizen.

Eine getypte Attribut-Wert-Matrix besitzt endlich viele Merkmale und einen Typ. Die Merkmale können atomar oder komplex sein. Eine Attribut-Wert-Matrix, die keine Merkmale beschreibt und den allgemeinsten Typ denotiert, ist das Analogon zur leeren Attribut-Wert-Matrix.

Bei dieser Darstellung ist zu beachten, dass die Typenbezeichnungen relativ zu dem definierten Typensystem zu sehen sind, d. h., dass die Typen extern – in einem definierten Typensystem – beschrieben sein müssen.

SUBSUMPTION VON GETYPTEN ATTRIBUT-WERT-MATRIZEN

Eine AWM D_σ subsumiert eine AWM D'_τ ($D_\sigma \sqsubseteq D'_\tau$) genau dann, wenn alle Informationen, die in D vorhanden sind auch in D' enthalten sind und darüber hinaus gilt ($\sigma \sqsubseteq \tau$). Dies bedeutet, dass alle von D'_τ beschriebenen Attribut-Wert-Strukturen auch von D_σ beschrieben werden.

Aufbauend auf dieser Beziehung wird die Unifikationsoperation definiert.

UNIFiKATION VON GETYPTEN ATTRIBUT-WERT-MATRIZEN

Das Ergebnis der Unifikation zweier getypter Attribut-Wert-Matrizen D'_σ und D''_τ ist die allgemeinste getypte Attribut-Wert-Matrix D_γ, die von D'_σ und D''_τ subsumiert wird. Wenn keine solche Struktur existiert, ist die Unifikation nicht definiert.

Im Anschluss an diese formalen Beschreibungen des Regelapparates soll beleuchtet werden, inwieweit die in der Linguistik und insbesondere in der

Computerlinguistik mit großem Gewinn angewendeten Technik der Unifikation für XML-Dokumente verwendet werden kann. Zuvor sollen jedoch, einerseits zur Exemplifikation des vorangegangenen Abschnitts, andererseits zur Darstellung dieser abstrakteren Informationsmodellierungsebene, gezeigt werden, wie Merkmalsstrukturen und deren Beschreibungen in SGML repräsentiert werden können.

4.2 SGML zur Repräsentation von Merkmalsstrukturen und deren Beschreibung

Die Text Encoding Initiative schlägt eine Annotationskonvention zur SGML-konformen Repräsentation von Merkmalsstrukturen bzw. der Merkmalsstrukturbeschreibungen vor. Ihre Verwendungsweise wird im Kapitel 16 der TEI-Guidelines (Sperberg-McQueen und Burnard 1994:475-519) vorgestellt. Die Entwickler dieses DTD-Moduls stellen auch mögliche alternativen Repräsentationsmöglichkeiten vor und rechtfertigen den von ihnen beschrittenen Weg (Simons und Langendoen, 1995).

Wie bereits ausgeführt, können Merkmalsstrukturen bzw. Attribut-Wert-Strukturen (AWS) auch dazu verwendet werden, Merkmalsstrukturen zu beschreiben. Die von den Entwicklern der Syntaxtheorie HPSG, vorgenommene Verdeutlichung der Trennung von Struktur und Beschreibung besteht in der Einführung der oben vorgestellten Notationskonvention. In den Veröffentlichungen zur HPSG werden Merkmalsstrukturen als Graphen und die Beschreibungen als Matrizen gedruckt.[42] Ein solcher Lösungsweg steht für eine SGML-basierte Modellierung natürlich nicht zur Verfügung, da eines der Hauptprinzipien von SGML in der Trennung von Inhalt und Form besteht. Die Unterscheidung in Graphen und Matrizen vollzieht sich ausschließlich auf der Ebenen der Form. In den Ausführungen zur TEI-basierten Merkmalsstrukturrepräsentation gibt es keinerlei Ausführungen zu diesem zentralen Problem der Arbeit in diesem Paradigma.

---

[42] A common source of confusion is that feature structures themselves can be used as descriptions of other feature structures. ... We choose to eliminate this possible source of confusion by using only totally well-typed, sort-resolved feature structures as (total) models of linguistic entities and AVM diagrams (not feature structures) as descriptions.

Pollard und Sag (1994:21)

Die TEI-DTD erlaubt lediglich die Modellierung von Merkmalsstrukturen. Da die Merkmalsstrukturen jedoch eine vielseitig verwendbare Datenstruktur sind und darüber hinaus ebendiese Datenstrukturen zur eigenen Beschreibung verwendet werden, ist eine wirklich zufriedenstellende Alternative auch schwer vorstellbar. Allerdings deutet die völlige Ignorierung des Problems der Unterscheidung von Modell und Beschreibung darauf hin, dass es von der TEI nicht zur Kenntnis genommen wurde. Es wäre z. B. zumindest denkbar gewesen, in den SGML-Strukturen einen Dokumentationsbereich vorzusehen, der den Zweck der Attribut-Wert-Struktur markiert, d. h. dient sie als Modell oder als Beschreibung,.

Da in den TEI-Richtlinien die Repräsentation von Merkmalsstrukturen unabhängig von ihrer Verwendung darlegt wird, muss auch hier auf die Unterscheidung bezüglich ihres Verwendungszwecks verzichtet werden. Da die konkreten Anwendungen allerdings mit den *Beschreibungen* von sprachlichen Einheiten befasst sind und Operationen, insbesondere die vorgestellte zentrale Operation Unifikation, ohnehin nicht auf die (totalen) Modelle angewendet werden können, liegt es nahe davon auszugehen, dass die Merkmalsstrukturen in erster Linie als Merkmalsstrukturbeschreibungsformalismus Verwendung finden.

Die folgende Beschreibung einer TEI-konformen Auszeichnung der Merkmalsstrukturen gibt diese nur sehr verkürzt wieder. Es soll lediglich die prinzipielle Herangehensweise dieses Ansatzes fokussiert werden, sowie auf mögliche in Ansätzen bereits existierende Alternativen eingegangen werden.

Eine Attribut-Wert-Struktur wird mit dem Element `<fs>` (mnemonisch verbunden zu feature structure) ausgezeichnet. Wenn sie keinen Inhalt besitzt, also leer ist, hat sie die Form `<fs></fs>`. Sie kann auch optional mit einer Typenangabe versehen werden, die dann mittels eines Attributs `type` vermerkt wird. Komplexere Strukturen besitzen Attribute, die mittels des Elementes `<f>` eingeführt werden können. Diesem Merkmal bzw. Attribut muss mit dem obligatorischen SGML-Attribut `name` eine Attributbezeichnung zugeordnet werden. Der Attributwert wird dann im Normalfall durch ein eingebettetes SGML-Element ausgedrückt. Wenn der Wert komplexer Gestalt ist, wird wiederum eine AWS durch das Element `<fs>` eingeführt, für atomare Werte sind insbesondere die leeren Elemente `<nbr>` (für Zahlen), `<str>` (für Zeichenketten), `<sym>` (für symbolische, potentiell extern definierte Werte) sowie `<plus>` und `<minus>` (für die boolschen Werte ja/nein bzw. true/false) von Relevanz.

Als Beispiel für eine Merkmalsstruktur mit einfachen, d. h. atomaren Attributwerten soll hier eine Merkmalsstruktur zur Beschreibung der grammatischen Spezifikation einer möglichen Lesart des deutschen Wortes ‚ihr' angegeben werden:

```
<fs type='wort'>
  <f name="ortho"><str rel="eq">ihr</str></f>
   <f name=Kasus><sym value=dativ></f>
   <f name=Genus><sym value=feminin></f>
   <f name=singular><plus></f>
</fs>
```

Das Attribut Singular erhält den boolschen Wert ‚plus', die Attribute Kasus und Genus erhalten die symbolischen Werte ‚dativ' und ‚feminin' und die orthografische Repräsentation der Wortform muss gleich (`rel=eq`) der Zeichenkette ‚ihr' sein.

Eine komplexe Merkmalsstruktur, d. h. eine Merkmalsstruktur, die als Attributwert wiederum eine Merkmalsstruktur besitzt, ist bei der Spezifikation des Lexikoneintrags des Wortes ‚sang' zu sehen:

```
<fs type='wort'>
   <f name=wortart><sym value=verb></f>
   <f name="ortho">
     <str rel="eq">sang</str>
   </f>
   <f name=zeitform><sym value=imperfekt></f>
   <f name=Kongruenz>
     <fs type='kong'>
       <f name=Kasus><sym value=nominativ></f>
       <f name=person><nbr value=3></f>
       <f name=singular><plus></f>
     </fs>
   </f>
</fs>
```

Bei dieser Merkmalsstruktur ist dem Attribut ‚Kongruenz' als Wert eine Attribut-Wert-Struktur vom Typ ‚kong' zugewiesen worden, welches bei einer Verarbeitung mit Hilfe der Unifikationsoperation auf Übereinstimmung mit der Kongruenzinformation anderer Wörter überprüft werden kann. Würde z. B. eine Unifikation mit der obigen AWS des Wortes ‚ihr' erfolgen, würde sie wegen der Kasusinkompatibilität fehlschlagen.

Das von der TEI erarbeitete Inventar zur Beschreibung von Merkmalsstrukturen ist von einer großen Liberalität geprägt. Dies hat gewisse Vorteile, die

insbesondere in ihrer variablen Anwendbarkeit liegen, allerdings erlaubt die fehlende Restriktivität des DTD-Moduls auch völlig illegitime Merkmalsstrukturen darzustellen. Beispielsweise können durch die TEI-DTD Merkmalsstrukturen repräsentiert werden, bei denen einem Merkmal mehrere Werte zugeordnet werden, z. B.:

```
<f name=person>
   <nbr value=3>
   <str value=drei>
   <sym value=third>
</f>
```

Es ist durchaus denkbar, dass eine derartige (Über-)Spezifikation in bestimmten Anwendungskontexten sinnvoll ist, jedoch verstößt sie gegen die Syntaxrestriktionen einer in der Linguistik verwendeten AWM (vgl. Abschnitt 4.1.2 und 4.1.3).

Um spezielle Syntaxrestriktionen überprüfen zu können, stellt die TEI einen separaten Mechanismus zur Verfügung mit Hilfe dessen es u. a. ermöglicht wird, genauere Angaben und Restriktionen über die Merkmalsstrukturen zu formulieren. Dieser Mechanismus trägt den Namen „Feature System Declaration (FSD)“ und wird in Kapitel 26 der TEI-Richtlinien beschrieben. Die FSD wird in einer separaten logischen Datei mit eigenem Dokumenttyp angelegt. Damit ist nicht mehr allein die Dokumentgrammatik des annotierten Dokuments für seine formale Gültigkeit verantwortlich. Wird eine Merkmalsstruktur unter der (fakultativen) Ausnutzung der Feature System Declaration annotiert, erfolgt zusätzlich zu der Minimalanforderung der syntaktischen Validität des Dokumentes bezüglich des TEI-DTD-Moduls für Merkmalsstrukturen auch ein weiterer Validierungsschritt, in dem das Dokument bezüglich der Restriktionen der FSD überprüft wird.

Eine sehr wichtige Anwendungsmöglichkeit dieser Methode bildet die am Beginn des Abschnitts 4.1.3 beschriebene Typisierung von Merkmalsstrukturen. Die im Normalfall vorzufindende Anwendung von Typen besteht darin, dass es die Typen sind, die die Spezifizierung von Merkmalen erst ermöglichen (well-typed) bzw. gar vorschreiben (totally well-typed). Werden Merkmalsstrukturen in der vorgestellten liberalen TEI-Form notiert, gibt es keine Möglichkeit, auf die vorhandenen Merkmale Einfluss zu nehmen.[43] Mit Hilfe

[43] Dies gilt bei der Verwendung der Schemasprache DTD. Andere Dokumentgrammatiken, insbesondere Schematron, nicht jedoch XSchema, erlauben die Spezifikation derartiger Restriktionen in der Dokumentgrammatik (vgl. allgemein zu verschie-

der Feature System Declaration können die möglichen Inhaltselemente eines Elementes `<fs>` in Abhängigkeit vom im Attribut `type` angegebenen Merkmalsstrukturtyp spezifiziert werden.

Die Spezifikationsmöglichkeiten der FSD gehen aber noch weiter. So erlauben es z. B. die Elemente `<if>` und `<then>` Inhaltsmodelle konditional, insbesondere kontextabhängig, zu definieren. Dies zeigt, dass die FSD eine weitere syntaktische Restriktion für bestimmte Dokumentinstanzen, nämlich Dokumentinstanzen, die Merkmalsstrukturen repräsentieren, erlauben. Insbesondere hier zeigt sich, dass die FSD eine Schemasprache darstellt. Dies hat allerdings eine entscheidende Konsequenz: die Gültigkeit der entsprechenden TEI-Dokumente muss nicht mehr nur gegenüber der DTD, sondern auch bezüglich der FSD festgestellt werden, was das Vorhandensein spezialisierter Software voraussetzt:

> We presume that application software will be developed that can parse the FSD and use the Information in it to validate the integrity of the integrity of every feature-structure in a marked-up document.
>
> Simons und Langendoen (1995:202)

Retrospektiv betrachtet kann festgestellt werden, dass sich diese Annahme nicht erfüllt hat und die FSD – und damit auch die Merkmalsstrukturannotation mit TEI-Mitteln außerordentlich selten vorkommt. Heute, mehr als zehn Jahre nach der Gründung der TEI gibt es eine Reihe reger Anwendungen der TEI-DTD – Anwendungen der Feature System Declaration zählen jedoch (noch?) nicht dazu.

Einen alternativen Weg der Repräsentation von Merkmalsstrukturen mittels XML gehen Sailer und Richter (2001). Die von ihnen verwendete Möglichkeit wurde zwar auch von der TEI betrachtet, jedoch ist sie von dieser Organisation verworfen worden (s. u.). Die Idee besteht darin, die ohnehin vorhandene Dokumentgrammatik (normalerweise die DTD) für eine direkte, durch die Typen der Merkmalsstrukturen gesteuerte Restriktion verwenden. Eine naheliegende Möglichkeit, dies zu erreichen, besteht darin, Merkmale nicht als Attribute des Elementes <f> zu behandeln, sondern als eigene Elemente. Ein Merkmal Kongruenz würde dann nicht als `<f name=Kongruenz>` sondern als Element `<Kongruenz>` annotiert. Die Angabe der möglichen Attribute erfolgt dann durch die Definition des Inhaltsmodells des entsprechenden Ele-

---

denen Schemasprachen Abschnitt 2.4). Die FSD kann unter dieser Perspektive auch als eine spezialisierte Schemasprache betrachtet werden.

ments. Und es ist genau diese Vergehensweise, die zur Zurückweisung dieses Ansatzes durch die TEI geführt hat:

> The chief disadvantage of this solution is that it leads to an unbounded proliferation of tags. Every user of TEI markup would be required to modify to add a new <!ELEMENT ...> definition for each needed feature.
>
> Simons und Langendoen (1995:194)

Für Sailer und Richter ist das Problem der Erweiterbarkeit der DTD jedoch nicht in dem Maße gegeben, da der Gegenstandsbereich ihrer Modellierung in den Merkmalsstrukturbeschreibungen der HPSG (Pollard und Sag, 1994) besteht. In spezifischen Theorien, also auch in der HPSG, gibt es nur eine begrenzte Anzahl von Attributen und Werten, wodurch die Erweiterbarkeit zu einem sekundären Designziel wird. Nichtsdestotrotz wurde ein Ansatz skizziert, der die Erweiterbarkeit nicht völlig ignoriert und dadurch auch auf andere Theorien, die Attribut-Wert-Strukturen verwenden, übertragbar ist.

Es wird eine allgemeine XML-DTD bereitgestellt, die es erlaubt, XML-Dokumente zu erstellen, die ein spezifisches Merkmals- bzw. Typensystem definieren. Diese XML-Datei wird dann mittels einer generischen Transformation mit der Sprache XSLT (Clark 1999) in eine die entsprechenden Dokumente restringierende DTD überführt. Die automatisch generierten DTDs bestehen aus einem allgemeinen, d. h. merkmals- und typensystemunabhängigen, und einem spezifischen Teil. Diese DTD wird abschließend den XML-annotierten Dokumenten zugeordnet, wodurch diese validierbar sind.

4.3 Unifikation von XML-Dokumenten

Wie in den Ausführungen zur Unifikation gesehen, erlaubt es dieser Ansatz, Informationen aus verschiedenen Quellen zusammenzuführen, auf Kompatibilität zu überprüfen und zu vereinigen. Das Ergebnis dieser Operation beinhaltet – wenn zueinander kompatible Informationen vereinigt wurden – alle Informationen, die in mindestens einer der Quellen vorhanden waren. Dies wird anhand der Merkmalsstrukturbeschreibungen deutlich – je mehr beschreibende Informationen zusammengetragen werden, desto weniger Merkmalsstrukturen werden beschrieben. Während die leere AWM alle Merkmalsstrukturen beschreibt, denotiert eine maximal spezifische AWM nur noch eine Merkmalsstruktur.

Die Zusammenführung von Teilinformationen mit dem Ziel der Vereinigung der dort verteilt vorhandenen Informationen ist im Kontext der Attibut-Wert-basierten Formalismen nicht von Bedeutung. Die Kumulation von kompatibler Information spielt vielmehr in nahezu allen Gebieten eine Rolle, die strukturierte Informationen verwenden. Vor diesem Hintergrund sollen Möglichkeiten beschrieben werden, die es erlauben, Informationen aus SGML- und XML-annotierten Texten auf Kompatibilität zu untersuchen und gegebenenfalls Teilinformationen mittels Unifikation zu vereinigen.

Die nachfolgenden Diskussionen betrachten zwei separierte Fälle. In einem der Fälle ist es möglich, dass verteilt erstellte Teildokumente zu einem (Gesamt-)dokument zusammengefasst werden. In diesem Fall kann davon ausgegangen werden, dass die Dokumente derselben Dokumentgrammatik genügen und (partiell) unterschiedliche Primärinformationen versammeln. Im anderen Fall unterscheidet sich die sekundäre Datenebene, d. h. die Strukturierung, wohingegen die Ebene der Primärdaten bei den zu unifizierenden Dokumenten identisch ist – eine Operation, speziell definiert für die im Abschnitt 3.7 vorgestellte Annotationsmethode.

4.3.1 Unifikation von dokumenttypidentischen XML-Dokumenten

Lobin und Reinsch (1999) schlugen die Unifikation von partiellen Dokumenten vor. Hierbei werden die Informationen zweier Dokumente, die derselben Dokumentenklasse zugeordnet sind, d. h. dieselbe DTD verwenden, vereinigt.

Im ersten Teil der Vorstellung dieses Ansatzes wird gezeigt, wie die DTDs dergestalt verändert werden können, dass sie es erlauben, partielle Dokumente zu beschreiben, denen nicht nur Primärdaten, sondern auch strukturierte Komponenten fehlen. Wenn z. B. die HTML-DTD zur Annotierung zweier Teildokumente benutzt wird, wobei eines die im `<head>` befindliche Meta-Information und das andere den im `<body>` befindlichen Hypertext beinhaltet, kann die HTML-DTD dergestalt verändert werden, dass das Inhaltsmodell des Wurzelelementes von `<!ELEMENT html (head, body)>` auf `<!ELEMENT html (head?, body?)>` abgeändert wird, damit auch die partiellen Teildokumente bezüglich einer DTD gültig sind und dadurch konform zum SGML-Standard annotiert werden können.

Im XML-Paradigma ist es jedoch auch möglich, partielle Dokumente als wohlgeformte Dokumente aufzufassen und auf die Generalisierung der DTD zu verzichten. Diese Möglichkeit wurde jedoch von den Autoren nicht zur Sprache gebracht.

Der zentrale Aspekt der Unifikation besteht in der Vereinigung der Dokumentinstanzen. Dies wurde anhand der Auflistung erwünschter Resultate der Unifikation beispielhafter abstrakter Inhaltsmodelle verdeutlicht. Diese werden nachfolgend entsprechend der dort vorgenommenen Klassifikation wiedergegeben und diskutiert.

Die Unifikation für SGML- bzw. XML-Instanzen ist in der angegebenen Form nicht als kommutative Operation beschrieben. Auf diesen Punkt wird an den für ihn relevanten Stellen gesondert eingegangen. Gegeben ist bis dahin die in der diskutierten Algorithmusbeschreibung nicht erwähnte Grundannahme:

```
Es gilt nicht für alle möglichen Dokumentinstanzen:
unify(Doc-inst1, Doc-inst2) = unify(Doc-inst2, Doc-inst1)
```

Diese Annahme steht im Gegensatz zur Definition der Unifikation von Merkmalsstrukturbeschreibungen, da bei der letztgenannten die Reihenfolge der zu unifizierenden Einheiten irrelevant ist. Eine der Hauptursachen für diese Nichtübereinstimmung der Eigenschaften liegt in einem prinzipiellen Unterschied zwischen Attribut-Wert-Strukturen und annotierten Dokumenten. Während bei der formalen Repräsentation der ersten ein *ungeordneter* Baum, oder – bei der Verwendung von co-indizierter Merkmalsstrukturen (vgl. Abschnitt 4.1.2.1) – ein entsprechender Graph verwendet wird, ist das formale Repräsentationsinventar für XML-annotierte Dokumente ein *geordneter* Baum bzw. – bei ID-IDREF-Verweisen – ein *geordneter* Graph.

Die Unifikation von dokumenttypidentischen XML-Dokumenten kann in simple und komplexe Fälle unterteilt werden. Als trivial kann die Unifikation eines vorhandenen Elementes mit einem nichtvorhandenen Element eingestuft werden. Ebenso trivial ist die Vereinigung zweier gleicher Elemente:

```
unify(<A>X</A>, Ø)         = <A>X</A>
unify(<A>X</A>, <A>X</A>) = <A>X</A>
```

Das Ergebnis der Unifikation zweier verschiedener Inhaltsmodelle wird wie folgt beschrieben:

```
unify(<A>X</A>, <B>Y</B>) = <A>X</A><B>Y</B>, wenn (A, B) im
                                              Inhaltsmodell
```

Allerdings sollte – auch wenn die Unifikation nicht als kommutative Operation aufgefasst wird – analog zu diesem Ergebnis auch die umgekehrte Reihenfolge erlaubt sein, wenn diese durch das Inhaltsmodell legitimiert ist. Konkretisierung:

```
unify(<A>X</A>, <B>Y</B>) = <A>X</A><B>Y</B>, wenn (A, B) im
                                              Inhaltsmodell
                 oder       <B>Y</B><A>X</A>, wenn (B, A) im
                                              Inhaltsmodell
                 sonst schlägt unify(<A>X</A>, <B>Y</B>) fehl.
```

Wenn die Operation jedoch als kommutativ aufgefasst würde, ergäbe sich ein weiteres Problem: nicht definiert und nicht entscheidbar wäre dann der Fall, wenn das (SGML-)Inhaltsmodell `(A & B)` oder das XML-Inhaltsmodell die Form `(A?, B, A?)` besitzt. Für derartige Fälle müsste die Unifikationsoperation um einen Algorithmus zur Prioritisierung der Elemente einer Menge gleichwertiger Ergebnisse ergänzt werden.

Ein weiterer problematischer und nicht diskutierter Fall besteht darin, dass weder (A, B) noch (B, A) durch das Inhaltsmodell des potentiellen Ergebnisdokumentes legitimiert sind und dennoch eine Unifikation plausibel erscheint. Zur Verdeutlichung dieses Problems sei (einmal mehr) HTML herangezogen: Wird z. B. versucht, das Dokument a mit dem Dokument b zu vereinigen, wäre das Ergebnis nicht definiert.

Dokument a:

```
<html><head><title>Beispiel Teil1</title>
              <META name="Author" content="Neil Example">
            </title></head>
       <body>...</body></html>
```

Dokument b:

```
<html>
     <META name="Date" content="Fall 2001">
     <body>...</body>
</html>
```

Es wird überprüft, ob für ein Inhaltsmodell des Elementes `<html>` die Form `(head, meta, body)` oder `(meta, head, body)` definiert wurde. Da dies in der HTML-DTD nicht vorgesehen ist, schlägt die Unifikation fehl. Das erwartete und erwünschte Ergebnis (Dokument c) wird nicht aufgebaut.

Dokument c:

```
<html><head><title>Beispiel Teil1</title>
               <META name="Author" content="Neil Example">
               <META name="Date" content="Fall 2001">
            </title></head>
      <body>...</body>
</html>
```

Andererseits stellt sich bei der Zulassung derartiger Unifikationen in einem noch stärkeren Maße das Problem der Prioritisierung, da die Unifikation dann gegebenenfalls nicht mehr nur Abfolge der Elemente, sondern auch die Einbettungen beinhalten müsste.

Der letzte von Lobin und Reinsch diskutierte Fall der Unifikation von Inhaltsmodellen macht implizit von einer Prioritisierung Gebrauch.

```
unify(<A>X</A>, <A>Y</A>) = <A>unify(X, Y)</A>
                            wenn unify(X, Y) zu einem
                            Ergebnis führt,
                            sonst
unify(<A>X</A>, <A>Y</A>) = <A>X</A><A>Y</A>,
                            wenn durch das Inhaltsmodell
                            legitimiert
sonst schlägt unify(<A>X</A>, <A>Y</A>) fehl
```

Eine Prioritisierung erfolgt im Falle des Zutreffens der Bedingungen für beide Regeln. Sie ergibt sich durch die Reihenfolge der gegebenen Regeln und die Semantik des Wortes ‚sonst'.

Die Unifikation von Dateninhalten wird ebenfalls in einer sehr einfachen, jedoch auch restriktiven Form definiert.

```
unify(a, Ø)  = a
unify(a, a)  = a
unify(a, ab) = ab
```

Das Ergebnis der Unifikation einer Zeichenkette bzw. eines Strings mit einer leeren Zeichenkette Z und das der Unifikation zweier identischer Strings Z ist die Zeichenkette Z. Der dritte Fall erscheint auf den ersten Blick unnötig restriktiv. Die Regel besagt, dass die Unifikation einer Zeichenkette Z1 mit dem String Z dann zu einem Ergebnis führt, wenn sich Z aus der Konkatenation der Zeichenkette Z1 und einem beliebigen String Z2 zusammensetzt, d. h. mit anderen Worten wenn Z mit Z1 beginnt. Es liegt bei dieser Definition die Frage nahe, warum nicht beliebige Teilzeichenketten bzw. Substrings zugelassen sind. Die Antwort lautet: die Definition erlaubt genau dies. Durch die vorherige Anwendung der ersten Regel auf den initialen – nicht mit Z1 identischen – Substring wird verhindert, dass die Zeichenkette Z mit Z1 beginnen muss.

Eine deutlichere Formulierung dieser Regel wäre jedoch wünschenswert. Konkretisierung:

```
unify(a, Ø)  = a
unify(a, xay) = xay , Wobei x und y beliebige, z. B. auch leere
                      Zeichenketten denotieren
```

Der Fall der Unifikation identischer Zeichenketten wird dadurch abgedeckt, dass sowohl der String x als auch der String y gleich der leeren Zeichenkette ist.

Bei der Definition der Unifikation von Dateninhalten fällt ein Gegensatz zur Verschmelzung von Elementinhalten auf: Falls versucht wird, verschiedene Dateninhalte zu unifizieren, scheitert die Operation; werden unterschiedliche Elementinhalte unifiziert, resultiert daraus eine Konkatenation der Daten, wenn das entstehende Inhaltsmodell durch die DTD legitimiert ist. Dies soll anhand eines Beispiels verdeutlicht werden:

Dokument a:

```
<!DOCTYPE text [<!ELEMENT text   (p+)>
                <!ELEMENT p      (#PCDATA)>]>
<text>
  <p>Allgemeine Information</p>
</text>
```

Dokument b:

```
<!DOCTYPE text [<!ELEMENT text  (p+)>
                <!ELEMENT p     (#PCDATA)>]>
<text>
  <p>Spezielle Information</p>
</text>
```

Die beiden unterschiedlichen Ergebnisse der Operation ‚unify' angewendet auf die Dokumente a und b sind entsprechend:

unify(dok-a,dok-b):

```
<!DOCTYPE text [<!ELEMENT text  (p+)>
                <!ELEMENT p     (#PCDATA)>]>
<text>
  <p>Allgemeine Information</p>
  <p>Spezielle Information</p>
</text>
```

unify(dok-b,dok-a):

```
<!DOCTYPE text [<!ELEMENT text  (p+)>
                <!ELEMENT p     (#PCDATA)>]>
<text>
  <p>Spezielle Information</p>
  <p>Allgemeine Information</p>
</text>
```

Werden jedoch unterschiedliche Dateninhalte unifiziert, scheitert die Operation. Die beiden Ausgangsdokumente zur beispielhaften Verdeutlichung sind jetzt:

Dokument a:

```
<!DOCTYPE text [<!ELEMENT text  (#PCDATA)>]>
<text>Allgemeine Information. </text>
```

Dokument b:

```
<!DOCTYPE text [<!ELEMENT text  (#PCDATA)>]>
<text>Spezielle Information. </text>
```

Die Operation ‚unify' scheitert, wenn sie auf die beiden Dokumente a und b angewendet wird, da keiner der beiden (nichtidentischen) textuellen Inhalte ein Substring des anderen Strings ist. Eine – vom Inhaltsmodell `PCDATA` abgedeckte – Textsequenz „string1 string2" ist kein mögliches Ergebnis.

Der letzte betrachtete Fall betrifft die Unifikation von Attributen. Die oben vorgenommene Einteilung in triviale und komplexe Fälle sei auch hier wieder aufgegriffen. Die trivialen Fälle sind:

```
unify(<A S="V">, <A S="V">) = <A S="V">
unify(<A S="V">, <A>)       = <A S="V">
```

Werden identische Attribut-Wert-Zuordnungen von Elementen vereinigt, ergeben sich keine Veränderungen. Die zweite Zeile bringt zum Ausdruck, dass nichtvorhandene Attribute auf die Unifikation keinen Einfluss haben. Die komplexeren Fälle sind folgendermaßen definiert:

```
unify(<A S="V">, <A T="W">) = <A S="V" T="W">,
                                 wenn S und T durch die DTD als
                                 Attribute legitimiert sind
unify(<A S="V">, <A S="W">) = <A S="unify(V, W)">
```

Sind zwei unifizierbare Elemente, die unterschiedliche Attribute besitzen, die Argumente der Operation `unify`, dann sollte das Ergebnis alle unterschiedlichen Attribute der Argumente mit den gegebenen Werten erhalten. Die geschieht jedoch nur für den Fall, dass die Attribute durch die DTD legitimiert wurden – eine überflüssige Einschränkung, wenn die zu unifzierenden Dokumente und das Ergebnisdokument Instanzen derselben DTD sind. Der letzte mögliche Fall betrifft die Unifikation von gleichen Elementidentifikatoren, gleichen Attributnamen und deren Werten. Ist dies gegeben, dann werden die Attributwerte unifiziert. Diese Regel lässt die oben aufgeführte Regel zur Unifizierung identischer Attribut-Wert Spezifikationen redundant werden.[44] Für nichtidentische Attributwerte kann die Operation allerdings exakt formuliert werden, da das gewünschte Resultat der Unifikation von Attributwerten in Abhängigkeit vom Typ dieses Wertes betrachtet werden muss. Der von Lobin und Reinsch beschriebene Fall zieht eine Parallele zur Unifikation von Dateninhalten. Diese Analogie kann jedoch nur für den Attributtyp CDATA gelten. Eine Erweiterung, die das Verhalten der Unifikation aller Typen einschließt, sei nachfolgend zusammengestellt:

[44] Die Regel `unify(<A S="V">, <A S="V">) = <A S="V">` ist ein Spezialfall der Regel `unify(<A S="V">, <A S="W">) = <A S="unify(V, W)">`, da das Ergebnis der allgemeineren Regel, angewandt auf zwei gleiche Attributwerte, `<A S="unify(V, V)">`, nach Ausführung von der eingeschlossenen Unifikation `<A S="V">` ist.

```
unify(<A S="V">,
      <A S="W">)
         = <A S="V">,      wenn W ein Substring von V und
                           S vom Typ CDATA
         = <A S="W">,      wenn V ein Substring von W und
                           S vom Typ CDATA
         = <A S="V  W)">, wenn V ≠ W und S vom Typ IDREFS,
                           ENTITIES oder NMTOKENS;
         = kein Ergebnis, wenn V ≠ W und S vom Typ ID,
                           ENTITY oder NMTOKEN bzw. wenn
                           S als Auswahlliste definiert
                           wurde.
```

Bei dieser Definition wurde darauf verzichtet, Attributtypkonversionen vorzunehmen. Dies könnte durchaus sinnvoll sein, z. B. würde es eine Typänderung von `IDREF` nach `IDREFS` ermöglichen, zwei verschiedene `IDREF`-Werte unifizieren zu können. Es wird hier allerdings davon ausgegangen, dass während des Prozesses der primären Informationsmodellierung – der Festlegung der DTD – die Attributtypfestlegung wohl durchdacht erfolgte, d. h. für das Beispiel, dass es beabsichtigt ist, einen 1 zu 1-Verweis - und nicht einen 1 zu n-Link – zu realisieren.

Bei der Vereinigung der Attribute ergibt sich das Thema ‚Kommutativität' nur an einer Stelle: bei der Vereinigung von Dateninhalten spielt, wie bei der Unifikation von `PCDATA`, die Reihenfolge der zu vereinigenden Zeichenketten eine Rolle. Sind die Attributwerte als Anhäufung (`IDREFS`, `ENTITIES` oder `NMTOKENS`) definiert, ist die Reihenfolge der Werte irrelevant, wodurch auch die Reihenfolge des Aufrufs der zu unifizierenden Dokumente keinen Einfluss auf die Bedeutung des Ergebnisdokumentes besitzt. Auch die Reihenfolge der Angabe der einem Element zugeordneten Attribute ist belanglos, da – wie bei den Attribut-Wert-Strukturen – alle angegebenen Attribute Gültigkeit besitzen. Es bleibt festzuhalten, dass diese Analogie nur für die Attribute gilt. Ein Element kann 0-n Attributbezeichner besitzen, die alle unterschiedlich benannt sein müssen. Die DTD stellt den Rahmen hierfür bereit und besitzt unter dieser Sichtweise einen vergleichbaren Status wie ein Typ einer getypten Attribut-Wert-Struktur, der die möglichen Attribute einer AWM definiert. Dadurch unterscheiden sich XML-Elemente bzw. Inhaltsmodelle von den Attributen entscheidend.

Als Konsequenz aus der Reihenfolgenabhängigkeit der Elemente ergibt sich für die Unifikation partieller Dokumentinstanzen ein weiteres gravierendes Problem, welches anhand eines Beispiels beschrieben wird:

Dokument a:

```
<!DOCTYPE text [<!ELEMENT text  (p+)>
                <!ELEMENT p     (#PCDATA)>
                <!ATTLIST p id  ID    #IMPLIED
                            txt CDATA #IMPLIED>]>
<text>
  <p>Text1</p>
  <p id='id-73'>Text2</p>
</text>
```

Dokument b:

```
<!DOCTYPE text [<!ELEMENT text  (p+)>
                <!ELEMENT p     (#PCDATA)>
                <!ATTLIST p id  ID    #IMPLIED
                            txt CDATA #IMPLIED>]>
<text>
  <p id='id-73' txt='Text automatisch generiert'></p>
</text>
```

Das intuitiv naheliegende Ergebnis und das Resultat der Unifikation unterscheiden sich: während erwünscht ist, dass die Abschnitte mit demselben Identifikator unifiziert werden, lautet das Ergebnis von `unify(dok-a,dok-b)`:

```
<!DOCTYPE text [<!ELEMENT text  (p+)>
                <!ELEMENT p     (#PCDATA)>
                <!ATTLIST p id  ID #IMPLIED
                            txt CDATA #IMPLIED>]>
<text>
  <p id='id-73' txt='Text automatisch generiert'>Text1</p>
  <p id='id-73'>Text2</p>
</text>
```

Dieses Ergebnis ist nicht nur nicht erwünscht, sondern es verstößt auch gegen die Gültigkeitsrestriktion, dass Attributwerte vom Typ ID im Dokument eindeutig sein müssen. Auch wenn für dieses Beispiel ein seltenerer und etwas extremer Fall ausgesucht wurde, kommt das grundlegende Entscheidungsproblem sehr häufig vor: Welches Element wird gewählt, wenn mit einem Element mehrere Elemente erfolgreich unifiziert werden können?

Als eine Möglichkeit bietet sich für dieses *matching*-Problem eine Prioritisierung der möglichen Unifikationspartner an. In diesem Prozess müsste, falls mehrere Kandidaten zur Verfügung stehen, eine Kennziffer für die Ähnlichkeit ermittelt werden, die entscheidet, welche Argumente die Unifikationsoperation erhält. Bei der Hinzufügung dieser Funktionalität entfernt sich der Unifikati-

onsprozess allerdings noch weiter von einer deklarativen Definition der Operation ‚Unifikation'.

Eine andere Möglichkeit zur Lösung kann darin bestehen, von vornherein zu verhindern, dass dieses Problem entsteht. Das kann erreicht werden, wenn dafür gesorgt wird, dass die Dokumentinstanzen „strukturähnlich" sind.

STRUKTURÄHNLICHKEIT

Zwei Dokumentinstanzen sind strukturähnlich, wenn alle Dateninhalte (`#PCDATA`) und alle Attribute der Dokumentinstanzen herausgefiltert wurden und die Baumrepräsentation der datenreduzierten Dokumente identisch ist.

Die Dokumenteninstanzen

```
<a><b Arg1='attwert'>text 1b1</b><b><c/> text 1b2</b></a>
```

und

```
<a id='a2'><b>text 2b1</b><b> text 2b2 <c/></b></a>
```

sind strukturähnlich und können der Unifikationsoperation zugeführt werden, ohne dass das matching der passenden Elemente zu Problemen führt. Jedes Element ist klar durch seine Position im Baum gekennzeichnet, was zur Folge hat, dass nur noch der Dateninhalt und die Attribute überprüft werden müssen. Eine solche Unifikation ist deklarativ. Die mathematischen Gesetze der Unifikation von der AWMs, d. h. das Kommutativitätsgesetz, das Assoziativitätsgesetz und das Idempotenzgesetz (vgl. Abschnitt 4.1.2.2), gelten auch für die derartig eingeschränkte Operation `unify(d,e)`. Diesen Vorteilen steht natürlich der Nachteil gegenüber, dass eine solche Restriktion die Anwendbarkeit der Operation extrem reduziert, da möglicherweise nur noch wenige Dokumente alle Voraussetzungen der Unifikation erfüllen. Die Situation kann möglicherweise dann verbessert werden, wenn die Dokumente automatisch in strukturähnliche Dokumente transformiert werden. Dies kann allerdings nur mit Heuristiken geschehen, die vergleichbar sind mit der oben – im Kontext der Lösungsstrategien des anderen Ansatzes – angesprochenen Prioritisierung.

Zusammenfassend kann festgestellt werden, dass die Unifikation von partiellen Dokumenten möglich ist, jedoch einige problematische Aspekte beinhaltet. Die Probleme sind auf eine Nichtübereinstimmung der Eigenschaft ‚Ordnung' zwischen Merkmalsstrukturen und SGML-Instanzen zurückzuführen. Hieraus folgt, dass die hier vorgestellte Operation nur unter

folgt, dass die hier vorgestellte Operation nur unter bestimmten Bedingungen ein eindeutiges Ergebnis erzielt.

4.3.2 Unifikation von primärdatenidentischen XML-Dokumenten

Ein bei der herkömmlichen Weise der Verwendung SGML-basierter Auszeichnungssprachen selten auftretendes potentielles Einsatzgebiet einer Informationsvereinigungsoperation bilden Dokumente, deren primäre Daten gleich sind, d. h. die sich nur in der Annotation unterscheiden. Ein Bedarf nach einer solchen Methode besteht jedoch.

Im Kapitel 1 wurden verschiede Möglichkeiten diskutiert, die gewährleisten sollen, dass textuelle Daten, die keine geordnete Struktur von Inhaltsobjekten bilden, mit den Mitteln von SGML repräsentiert werden können. Nach extensiver Betrachtung bereits verwendeter Möglichkeiten wurde festgestellt, dass diese Methoden diverse Mankos mit sich bringen. Aus diesem Grund wurde im Abschnitt 3.7 die Verwendung einer Technik vorgeschlagen, die zwar in der Literatur als prinzipiell zur Verfügung stehende Möglichkeit kurz erwähnt (Sperberg-McQueen und Burnard, 1994:755f.), jedoch bisher in der Forschungsliteratur noch nicht diskutiert wurde: die separate Annotation der Phänomene. Es wurde gezeigt, dass diese Methode eine Vielzahl von Vorzügen gegenüber anderen hierfür verwendeten Ansätzen besitzt. Der große Vorteil lässt sich auf die nur indirekt vorgenommene Verknüpfung und die damit verbundene vollständige Eigenständigkeit der Annotation zurückführen. Genau dieser Vorteil ist allerdings auch der Nachteil dieser Methode: Wenn die Annotationsebenen zueinander nicht in Relation gesetzt werden können und wenn die Dokumente nach erfolgter separater Annotation die Dokumente quasi unverbunden nebeneinander stehen, ist das Resultat der u. U. sehr aufwändigen Auszeichnungsprozesse von keinem großen Wert.[45] Die nachfolgend definierte Unifikation von primärdatenidentischen XML-Dokumenten hat das Ziel eine Methode bereitzustellen, um die dort vorhandenen impliziten Verknüpfungen zu explizieren. Dadurch wird der separaten Annotation der Nachteil genommen – es verbleiben die Vorteile.

Die Diskussion der Phänomene und der gewünschten Ergebnisse der Unifikation soll anhand des im Abschnitt 3.7 bereits verwendeten Beispielsatzes, der

[45] Vermutlich ist dies der Grund, warum bisher noch nicht vorgeschlagen wurde, die Methode der separaten Annotation zu verwenden.

jeweils mindestens den Status eines well-balanced Fragments besitzt[46], vorgenommen werden. Diese Fragmente seien hier nochmals in einer gekürzten Form widergegeben:

```
<!--  Sätze (<s>): -->
<s><s>die Aufnahme stoppt</s> und <s>das Deck schaltet sich aus</s></s>

<!--  Nominalphrasen (<np>): -->
<np>die Aufnahme</np> stoppt und <np>das Deck</np> schaltet sich aus

<!--  Syntaktische Funktionen Subjekt (<subj>): -->
<subj>die Aufnahme</subj> stoppt und <subj>das Deck</subj> schaltet sich aus

<!--  Verbalphrasen (<vp>): -->
die Aufnahme <vp>stoppt</vp> und das Deck <vp>schaltet sich aus</vp>

<!--  Silben (<syll>): -->
<syll>die</syll>
<syll>Auf</syll><syll>nah</syll><syll>me</syll><syll>stoppt</s...

<!--  Morphe (<m>): -->
<m>die</m> <m>Auf</m><m>nahm</m><m>e</m> <m>stopp</m><m>t</m> <m>und</m>
<m>das...

<!--  (vollständige) Wörter (<w>): -->
<w>die</w> <w>Aufnahme</w> <w>stoppt</w> <w>und</w> <w>das</w> <w>Deck</w>
scha...
```

In einem ersten Schritt werden nachfolgend exemplarisch mögliche und gewünschte Ergebnisse der Unifikation ausgewählter Ebenen diskutiert. Dabei können mit Hilfe dieser Beispiele mehrere generalisierbare Fälle unterschieden werden.

Fall a

Im Fall a werden die unproblematischen Fälle zusammengefasst.

Fall a1: Die Unifikation von zwei oder mehr Ebenen führt zu keinen weiteren Verschachtelungen. Dies gilt z. B. für die Ebenen `<np>` und `<vp>`, deren Unifikation zu folgendem Ergebnis führen sollte:

```
<!--  unify(Ebene <vp>, Ebene <np>): -->
<np>die Aufnahme</np> <vp>stoppt</vp> und <np>das Deck</np
> <vp>schaltet sich aus</vp>
```

Bisher nicht annotierte Einheiten aus XML-Fragmenten, die den Status well-balanced besitzen, erhalten gegebenenfalls die Annotationen der mit ihnen unifizierten Argumente.

[46] Vgl. hierzu Fußnote 32 auf Seite 66.

Fall a2: Die Unifikation von zwei oder mehr Ebenen führt zu einem Ergebnisdokument, das in der Form einer geordneten Hierarchie von Inhaltsobjekten strukturiert ist.

Die Unifikation der Ebenen <s> und <vp>, der Ebenen <s> und <np> sowie die komplexe Unifikation der Ebenen <s>, <np> und <w> sind Beispiele hierfür:

```
<!--  unify(Ebene <s>, Ebene <np>): -->
<s><s><np>die Aufnahme</np> stoppt</s> und <s
><np>das Deck</np> schaltet sich aus</s></s>

<!--  unify(Ebene <s>, Ebene <vp>): -->
<s><s>die Aufnahme <vp>stoppt</vp></s> und <s
>das Deck <vp>schaltet sich aus</vp></s></s>

<!--  unify(Ebene w, unify(Ebene <s>, Ebene <np>)): -->
<s><s><w>die</w> <w>Aufnahme</w> <w>stoppt</w></s>
<w>und</w> <s><w>das</w> <w>Deck</w> schaltet <w>sich</w>
aus</s></s>
```

Es ist zu sehen, dass die Anwendung der Unifikationsoperation hierbei zu regulär verschachtelten Umgebungen führt.

Fall b

Auch die im Fall b zu betrachtende Ausgangssituation führt zu wohlgeformten XML-Dokumenten. Allerdings kann dieser Fall nicht als unproblematisch bezeichnet werden, da beim Auftreten der hier thematisierten Konfiguration der zu unifizierenden Dokumente verschiedene Lösungsalternativen gerechtfertigt werden können.

Die Unifikation der Annotationsebenen der Subjekte <subj> und der Nominalphrasen <np> kann für den obigen Beispielsatz zu zwei verschiedenen, jeweils korrekt verschachtelten Umgebungen führen.

Lösungsalternative 1 und 2:

```
<!-- unify(Ebene <np>, Ebene <subj>): -->

<!-- Ergebnisalternative a: -->
<np><subj>die Aufnahme</subj></np> stoppt und <np
><subj>das Deck</subj></np> schaltet sich aus

<!-- Ergebnisalternative b: -->
<subj><np>die Aufnahme</np></subj> stoppt und <subj
     ><np>das Deck</np></subj> schaltet sich aus
```

Der identische Start- und Endpunkt der Umgebungen führt dazu, dass nicht automatisch festgelegt werden kann, ob Alternative a oder b gewählt wird. Es ist jedoch zu beachten, dass sich die Interpretationen der Ergebnisoptionen unterscheiden, wie deren Verbalisierung verdeutlicht: Alternative a bringt zum Ausdruck, dass die Nominalphrasen aus einem Subjekt bestehen, während b die (wesentlich sinnvollere) Aussage impliziert, dass das jeweilige Subjekt durch eine NP realisiert wird. Eine dritte Alternative besteht darin, dass keinerlei verschachtelte Umgebungen angegeben werden. Vielmehr soll ein Repräsentation verwendet werden, die die Interpretation erlaubt, dass `<np>` und `<subj>` gleichzeitig beginnen und enden. Hierzu würden sich u. a. die vom MATE-Projekt verwendeten und in Abschnitt 3.2.1 beschriebenen Verweistechniken eigenen.

Fall b - Lösungsalternative 3:

```
<!-- Ergebnisalternative c: -->

<np id='n1'>die Aufnahme</np> stoppt und <np id='n1'
>das Deck</np> schaltet sich aus

<subj-annotation>
     <subj idref='n1'/><subj idref='n2'/>
</subj-annotation>
```

Der qualitative Unterschied zu den Lösungsalternativen 1 und 2 ist deutlich zu sehen: die Elemente `<np>` und `<subj>` stehen nicht in einer Inklusionsbeziehung zueinander, ein Umstand, der den großen Vorteil besitzt, dass die einzige Interpretation darin besteht, dass die Umgebungen gemeinsam geöffnet und geschlossen werden. Allerdings besitzt dieser Lösungsweg genau die gravierenden Nachteile der indirekten Textauszeichnung, die ebenfalls schon im Abschnitt 3.2.1 thematisiert worden sind – Gründe, die mit dazu geführt haben diesen Annotationsansatz zurückzuweisen und im Abschnitt 3.7 eine

bessere Repräsentationsmöglichkeit zu entwickeln. Dieses mögliche Unifikationsresultat kann zwangsläufig nicht das erstrebenswerteste Ergebnis sein.

Im Abschnitt 3.6 wurde eine weitere Möglichkeit der Repräsentation verschiedener Ebenen beschrieben. Die SGML-Architekturen besitzen zwar eine eng begrenzte Mächtigkeit, stellen aber exakt für die im Fall b beschrienen Daten eine ideale Modellierung zur Verfügung. Zu beachten ist hierbei jedoch, dass die architektonischen Beziehungen für DTDs, d. h. für Klassen von Dokumenten angegeben werden. Sie stellen somit Generalisierungen dar, die zwar einerseits der Wunsch einer jeden Informationsmodellierung sind, andererseits jedoch müssen die Generalisierungen valide sein, eine Herausforderung, die durch eine Einzelfallbetrachtung nicht gelöst werden kann.

Fall c

Der letzte und komplizierteste Fall, betrifft die Unifikation zweier Dokumente, deren naive (oder auch intuitive) Lösungen zu Dokumenten führt, die nicht wohlgeformt (oder auch nur well-balanced) sind, da sich ihre Umgebungen überschneiden.

Fall c – nicht XML-konforme Lösungsalternative: Die Unifikation der Ebenen `<m>` und `<syll>` könnte zu der inkorrekt verschachtelten Dokumentinstanz führen, die partiell bereits auf Seite 65 abgebildet war:

```
<-- Denkbares Unifikationsergebnis von <m> und <syll>, -->
<-- welches nicht der XML-Syntax genügt:                -->

<syll><m>Die</m></syll> <syll><m>Auf</m></syll
><syll><m>nah</syll><syll>m</m><m>e</m></syll
> <syll><m>stopp</m><m>t</m></syll
> <syll><m>und</m></syll> <syll><m>das</m
></syll
> <syll><m>Deck</m></syll
> <syll><m>schal</syll><syll>t</m><m>et</m></syll
> <syll><m>sich</m></syll
> <syll><m>ab</m></syll>
```

Selbstverständlich ist auch hier die bereits im Fall b beschriebene Problematik virulent, d. h., es ist nicht befriedigend, dass in dieser Annotation die Interpretation nahe gelegt wird, dass die Silben Morphe enthalten. Die gravierendere Problemstellung hingegen besteht darin, dass geklärt werden muss, wie auf den Verstoß gegen die XML-Syntax reagiert werden soll. Wenn Ergebnisdokumente, die nicht XML-konform sind, verhindert werden sollen, bieten sich zwei Lösungen an: Die Operation Unifikation scheitert oder das Ergebnis wird

in eine zum intendierten Resultat kompatible Repräsentation, die den XML-Restriktionen genügt, überführt. Da für beide dieser Lösungen Szenarien existieren, die ihr jeweiliges Resultat wünschenswert erscheinen lassen, werden hier beide vorgestellt. Um die Unterscheidbarkeit der Operationen zu erlauben, werden sie als Unifikation erster und Unifikation zweiter Stufe bezeichnet. Beide Unifikationsdefinitionen sind so weit gefasst, dass sie die Fälle a und b beschreiben ohne Lösungsalternativen zu favorisieren.

UNIFIKATION PRIMÄRDATENIDENTISCHER XML-INSTANZEN (ALLGEMEIN)

Zwei primärdatenidentische XML-Dokumentinstanzen D1 und D2 können zu einem Dokument E unifiziert werden. Das Dokument E ist das allgemeinste Dokument, das alle Informationen, die D1 und die in D2 enthalten sind, beinhaltet.

Das Ergebnis E beinhaltet somit die gemeinsamen Primärdaten sowie alle Annotationen aus D1 und D2. Es ist zu beachten, dass eine derartige Definition nicht festlegt, dass das Ergebnis konform zur Syntax von XML sein muss. Auch das oben als „denkbares Unifikationsergebnis“ der Annotationsebenen Morph und Silbe nicht XML-konforme Dokument ist durch diese Unifikationsdefinition als Ergebnis legitimiert. Des Weiteren fällt auf, dass diese Definition das Resultat der Operation nicht auf ein Ergebnis beschränkt, vielmehr sind auch mehrere Ergebnisse, z. B. die verschiedenen Lösungsalternativen des Falls b durch die gewählte Formulierung legitimiert.

An der Vielzahl möglicher Lösungen kann gesehen werden, dass diese Definition, um eine praktikable Operation zur Verfügung zu stellen, weiter verfeinert werden muss. Ein erster problematischer Punkt besteht darin, dass die dritte Lösungsalternative des Falls b, trotz ihrer Mankos gleichberechtigt legitimiert wird. Der Unterschied zwischen den Alternativen eins und zwei auf der einen und drei auf der anderen Seite lässt sich daran festmachen, dass in den ersten beiden Optionen die Primärdateninklusion direkt (`PCDATA` innerhalb von Umgebungen) und in der dritten indirekt (Umgebungen verweisen auf Umgebungen die `PCDATA` beinhalten) erfolgt.

UNIFIKATION PRIMÄRDATENIDENTISCHER XML-INSTANZEN

Zwei primärdatenidentische XML-Dokumentinstanzen D1 und D2 können zu einem Dokument E unifiziert werden. Das Dokument E ist das allgemeinste Dokument, das alle Informationen, die D1 und die in D2 enthalten sind, beinhaltet. Für jedes Zeichen z aus der Menge der Pri-

märdaten ($z \in PD$) und jede Umgebung aus D1 und jede Umgebung aus D2 gilt: Wenn z zwischen einer umgebungsöffnenden und einer umgebungsschließenden Markierung der Ausgangsdokumente steht, dann steht z im Ergebnisdokument ebenfalls zwischen diesen Markierungen.

Durch diese verbesserte Unifikationsdefinition ist nun eine problematische Lösungsalternative ausgeschlossen, die Optionen eins und zwei bleiben weiterhin legitimiert. Es war oben angemerkt worden, dass das Hauptproblem dieser Alternativen darin zu sehen ist, dass sie an den Stellen im Dokument verschachtelte Umgebungen suggerieren, an denen Elemente an derselben Stelle beginnen bzw. enden. Bis auf weiteres kann dieser Punkt dadurch ausgeräumt werden, dass vermerkt wird, aus welchem Ausgangsdokument welche Annotation stammt. Dies kann z. B. mit Namensräumen (vgl. Abschnitt 2.5) erfolgen.

Besonders kritisch anzumerken ist jedoch der Umstand, dass – auch nach der Revision der Definition – weiterhin nicht XML-konforme Ergebnisse durch diese Operation legitimiert werden. Diesem Pumkt widmen sich die beiden folgenden spezielleren Operationsdefinitionen – mit unterschiedlichen Mitteln.

Basierend auf dem verfeinerten Begriff von Unifikation werden zwei verschiedene, alternative Stufen von Lösungen angeboten.

UNIFIKATION ERSTER STUFE

Zwei primärdatenidentische XML-Dokumentinstanzen D1 und D2 können zu einem Dokument D1'2' unifiziert werden. Das Dokument D1'2' ist das allgemeinste Dokument, das alle Informationen, die in D1 und in D2 enthalten sind, beinhaltet. Für jedes Zeichen z aus der Menge der Primärdaten ($z \in PD$), jede Umgebung aus D1 und jede Umgebung aus D2 gilt: Wenn z zwischen einer umgebungsöffnenden und einer umgebungsschließenden Markierung der Ausgangsdokumente steht, dann steht z in D1'2' ebenfalls zwischen diesen Markierungen. Ist D1'2' ein wohlgeformtes XML-Dokument, wird es in das Ergebnisdokument E, ist D1'2' kein wohlgeformtes XML-Dokument, ist die Operation gescheitert.

Wird sie also z. B. auf die obige Silben- und Morphannotation angewendet, gibt es kein Ergebnis. Die Unifikation erster Stufe schlägt fehl. Diese Definition führt dazu, dass sie als Test für die Kompatibilität verschiedener Annotationsebenen verwendet werden kann. Der Preis ist jedoch, dass sie – wird eine Inkompatibilität festgestellt – keine Lösung liefert. Die nun folgende Unifikati-

onsdefinition setzt hier an und stellt mit der Unifikation zweiter Stufe eine Lösung bereit.

UNIFIKATION ZWEITER STUFE

Zwei primärdatenidentische XML-Dokumentinstanzen D1 und D2 können zu einem Dokument D1'2' unifiziert werden. Das Dokument D1'2' ist das allgemeinste Dokument, das alle Informationen, die D1 und die in D2 enthalten sind, beinhaltet. Für jedes Zeichen z aus der Menge der Primärdaten (z ∈ PD) und für jede Umgebung aus D1 und D2 gilt: Wenn z zwischen einer umgebungsöffnenden und einer umgebungsschließenden Markierung der Ausgangsdokumente steht, dann steht z in D1'2' ebenfalls zwischen diesen Markierungen. Ist D1'2' ein wohlgeformtes XML-Dokument, wird es das Ergebnisdokument E, ist D1'2' kein wohlgeformtes XML-Dokument, wird es in ein XML-konformes Dokument E transferiert, wobei inkorrekt verschachtelte Umgebungen aus D1'2' durch die Fragmentierungstechniken entschachtelt werden.

Das Ergebnis dieser Operation sei jetzt wiederum an dem in dieser Diskussion verwendeten Beispiel der Markierung der Ebenen `<m>` und `<syll>` exemplifiziert.

```
<-- Unifikationsergebnis der Unifikation Zweiter Stufe-->
<-- angewedet auf die Ebenen <m> und <syll>           -->

<syll><m>Die</m></syll> <syll><m>Auf</m></syll
><syll><m id='i3' status='part' next='i4'>nah</m></syll
><syll><m id='i4' status='part'
          prev='i3'>m</m><m>e</m></syll
> <syll><m>stopp</m><m>t</m></syll> <syll
><m>und</m></syll> <syll><m>das</m></syll> <syll
><m>Deck</m></syll> <syll
><m id='i8' status='ic' next='i9'>schal</m></syll><syll
><m id='i9' status='part' prev='i8'>t</m><m>et</m></syll
> <syll><m>sich</m></syll> <syll><m>ab</m></syll>
```

Die durch die Operation fragmentierten Umgebungen sind zur besseren Übersicht hervorgehoben. Ebenfalls zur Verbesserung der Lesbarkeit wurde

hier nur die die Unvollständigkeit markierende Angabe aufgeführt (`status='part'`).[47]

Die Unifikationsoperation stellt durch die Aufteilung in verschiedene Stufen ein mächtiges Operationsinstrumentarium bereit, welches es einerseits in der zweiten Stufe erlaubt, separate Annotationen zur einer komplexeren Annotation zu vereinen, andererseits jedoch auch Dokumente durch die Anwendung der ersten Stufe auf kompatible Annotationen zu überprüfen und somit die Existenz inkompatibler Hierarchien automatisch herauszufinden. Bisher ist es jedoch noch nicht möglich, Beziehungen zwischen den verschiedenen Annotationen herzustellen.

[47] Die Definition der Unifikation zweiter Stufe legt nicht fest, welche Umgebungen aufgeteilt werden. Wie schon Fall (b) bei der Unifikation erster Stufe zu erkennen war, wird auch hier nicht angestrebt, eine eindeutige Lösung zu erzielen.

5 Architektur des Annotationssystems

Die Möglichkeiten zur Repräsentation von Daten, die (potentiell) die Strukturierung gemäß verschiedener Hierarchien erfordern, ist extensiv im Kapitel 1 diskutiert worden. Dort ist empfohlen, diese Daten durch die separate multiple Annotation derselben zu beschreibenden Ausgangsdaten – der Primärdatenebene – zu repräsentieren. Es soll an dieser Stelle nochmals explizit darauf hingewiesen werden, dass eine unterschiedliche Hierarchisierung auch dann angenommnen wird, wenn die verschiedenen Strukturierungen auch in einer Hierarchie ausgedrückt werden können.

Mit den aus der Linguistik stammenden Verfahren der Unifikation können die Annotationen der separaten Daten auf Kompatibilität überprüft und – je nach angenommener Verarbeitungsstringenz – gegebenenfalls zusammengeführt werden (vgl. Abschnitt 4.3.2). In diesem Kapitel soll für diesen neuen Weg der verteilten Informationsrepräsentation ein Modell entwickelt und dieses in Form einer Architekturdarstellung vorgestellt werden (Abschnitt 5.1).

Im Abschnitt 5.2 wird dann ein neuerer Vorschlag zur Wissensrepräsentation annotierter Daten vorgestellt und auf die hier vorgeschlagene Repräsentationsarchitektur angepasst.

Im Abschnitt 5.3 wird gezeigt, dass dieses Modell völlig neue Perspektiven für die Strukturierung von Dokumentgrammatiken bietet.

5.1 Komponenten und Prozesse der Repräsentation

Das Modell der Repräsentation komplex strukturierter Texte besitzt in seinem Kern die zu strukturierenden Daten selbst. Diese Daten können gemäß spezieller Dokumentgrammatiken annotiert werden. Im einfachsten Fall gibt es eine Dokumentgrammatik und eine Annotation, die in einem Dokument vereint sind, d. h. alle SGML-Dokumente und alle gültigen XML-Dokumente sind in diesem Modell repräsentiert. In Ergänzung zu dieser Standardrepräsentation kann jedes Dokument gemäß weiterer Dokumentgrammatiken strukturiert werden. Es gibt keine Beschränkung der maximalen Anzahl von Annotationen, das Modell erlaubt die stetige Vergrößerung der verwendeten Annotationen, d. h. es ist ein offenes Modell. Die annotierten Textdaten bilden eine Verbindung zwischen den unterschiedlichen Annotationsebenen.

Die Informationsaufbereitung entsprechend der hier vorgestellten Architektur des Annotationssystems betrifft verschiedene Komponenten der Textannotation:

- die Datenaufbereitung
- die Dokumentgrammatik bzw. meist die Dokumentgrammatiken
- den Annotationsprozess

Die Datenaufbereitung muss dergestalt erfolgen, dass eine separate Annotation der relevanten Phänomene möglich ist. Dieser Prozess und die dafür zu beachtenden Restriktionen sind im Abschnitt 3.7 ausführlich beschrieben worden, so dass hier nur die Kernpunkte der Datenaufbereitung in einer prägnanten Darstellung zusammengefasst werden sollen.

Die zu annotierenden Daten werden in einem ersten Schritt klassifiziert. Daten können einen Primärdatenstatus oder einen Metadatenstatus besitzen. Die primäre Datenebene enthält den Beschreibungsgegenstand – d. h. die die eigentlichen Informationen beinhaltenden und noch nicht annotierten Texte – während die Metadaten Informationen über diese Daten beinhalten. In bereits annotierten Texten sind die Auszeichnungen bzw. das Mark-Up ein klassisches Beispiel für Metadaten, jedoch können auch Textdaten selbst Einheiten enthalten, die als Metadaten klassifizierbar sind. Ein Beispiel hierfür findet sich in nahezu allen technischen Dokumentationen: bestimmte Informationen werden mittels extern definierter Schüsselwörter, wie ‚Hinweis', ‚Warnung' etc., eingeleitet. In einer inhaltsbezogenen Datenaufbereitung werden diese Schlüsselwörter als Metadaten klassifiziert.

Die inhaltliche Klassifikation erlaubt die für das Annotationssystem elementare technische Repräsentation der Texte. Die in einem intellektuellen Prozess[48] den Primärdaten zugerechneten Textteile werden in den unterschiedlichen, separaten Annotationen als Primärdaten in einem technischen Sinne (vgl. Definition auf Seite 65) aufbereitet. Diese Primärdaten sind einerseits der Gegenstandsbereich für die potentiell mehrfachen Auszeichnungen, andererseits bilden sie das Bindeglied zwischen den Annotationsebenen.

Die Verwendung der Primärdaten zur Verknüpfung der Ebenen hat zur Folge, dass ambig gebrauchte Zeichen, insbesondere Zeichen, die auch als Trennzeichen verwendet werden (z. B. Leerzeichen), über alle separaten Annotati-

[48] Diese Aufgabe besitzt einen ähnlichen Status wie der Dokumenttypanalyseprozess während des DTD-Designs (vgl. Maler und El Andaloussi, 1996).

onen hinweg identisch repräsentiert werden müssen. Die Identität kann auch durch genauere Spezifizierungen von Äquivalenzklassen erreicht werden, ein Punkt, der bereits ausführlicher diskutiert wurde (vgl. S. 70 f.).

Die Anzahl der Annotationsebenen ist nicht begrenzt, weder per se durch die hier vorgestellte Architektur noch de facto durch Entscheidungen, die am Beginn des Informationsmodellierungsprozesses getroffen werden. Die Relevanz dieses Punktes wird besonders deutlich, wenn die Genese einiger der derzeit standardisierten Dokumentgrammatiken, z. B. die HTML-DTD, genauer betrachtet wird. Werden die DTD-Versionen verglichen, zeigt sich, dass innerhalb von wenigen Jahren (mindestens) drei Versionen (HTML 2.0, HTML 3.2 HTML 4.01) ein weithin anerkanntes Annotationsschema darstellten. Allein hieran zeigt sich, dass eine Erweiterbarkeit der Dokumentgrammatiken ein Designziel darstellen sollte, wobei die Erweiterbarkeit nicht notwendigerweise mit einer allgemeinen Relevanz oder gar Akzeptanz verbunden sein muss, da auch Erweiterungen möglich sein sollen, die für bestimmte, seltene Dokumentarten relevant sind. Auch dies ist in – den neueren Versionen von – HTML (und den meisten anderen standardisierten Dokumentgrammatiken) möglich, allerdings nur über den Umweg der Verwendung allgemeiner Attribute wie z. B. `html:class` oder `html:type`. Eine elaborierte Dokumentmodellierung ist allerdings nicht über Umwege, sondern nur direkt möglich.

Nach der Etablierung der Primärdaten muss im Normalfall für jedes zur Beschreibung vorgesehene bzw. auserwählte Phänomen eine eigene Dokumentgrammatik entwickelt bzw. bereitgestellt werden. Hiervon kann abgewichen werden, wenn sinnvolle Kombinationen von Phänomenen herausgearbeitet wurden, die in einer Dokumentgrammatik modelliert werden sollen.

Auch das Design der Dokumentgrammatik muss in Rechnung stellen, dass die Annotationsebenen durch die Primärdaten verknüpft werden. Dies hat eine unmittelbare Konsequenz für den Modellierungsprozess: Sollten bestimmte Primärdaten irrelevant für eine Beschreibungsebene sein, müssen diese Daten dennoch als Primärdaten in der Annotation vorhanden sein, wobei die technische Definition der Primärdaten maßgebend ist[49]. Diesem scheinbaren Widerspruch kann dadurch begegnet werden, dass für diese –

[49] Aus der Perspektive der zu annotierenden Phänomene sind diese Daten natürlich nicht primär, vielmehr könnten sie u. U. weggelassen werden. Z. B. kann bei einer Annotation eines gesprochenen Diskurses, deren einziger Zweck darin besteht, einer semantischen Analyse als Grundlage zu dienen, auf die Transkription von selbst korrigierten Versprechern verzichtet werden.

bzgl. des Phänomens – irrelevanten Primärdaten eine spezielle Markierung in der Dokumentgrammatik vorgesehen wird, die es erlaubt, die entsprechenden Passagen in einem technischen Sinne als Primärdaten zu repräsentieren. Eine derartige Auszeichnung kann z. B. durch die Verwendung eines Elementes `<ignore>` erfolgen. Bei der Interpretation der ausgezeichneten Daten werden dann die entsprechend markierten Abschnitte herausgefiltert oder nicht beachtet. Eine ähnliche Lösung wurde auch in einem bereits erwähnten[50] Dokumentmodellierungsvorschlag von Durand et al. (1996) angesprochen, wobei dort diese Klassifizierung auf einer abstrakteren Repräsentationsebene – den streams – erfolgen sollte.

Die verwendeten Dokumentgrammatiken werden auf die zu annotierenden Daten angepasst. Die Vielzahl der zur Verfügung stehenden Dokumentgrammatiken wird in einem Dokumentgrammatikpool zusammengestellt. Dieser Pool stellt ein abstraktes Gebilde dar. Dieses theoretische Konstrukt erhält seine faktische Realisierung, indem alle zur Verfügung stehenden Dokumentgrammatiken wie auch immer verfügbar sein sollten. Die Referenzierung der benötigten Dokumentgrammatiken kann durch die Angabe eines Namensraums erfolgen (vgl. Absatz 2.5). Die Wahl der Schemasprache ist nicht von Bedeutung, es ist sogar völlig irrelevant, ob diese Schemasprache bereits standardisiert ist oder ob sie jemals normiert wird. Die Ansammlung bzw. Zusammenstellung von Dokumentgrammatiken kann als eine Nebeneinanderstellung gleichberechtigter Einheiten aufgefasst werden. Die Dokumentgrammatiken können jedoch auch u. U. in einer Hierarchie strukturiert werden, ein Punkt, der im Abschnitt 5.3 besprochen wird. Für den Annotationsprozess ist es unbedeutend, wie die Ansammlung strukturiert ist.

Für den Prozess der Annotation gilt es insbesondere zu beachten, dass die Primärdaten als Verknüpfung verwendet werden. In den Fällen, in denen die Dokumentgrammatiken eine Wahlmöglichkeit einräumen, welche Daten wie annotiert werden sollen[51] oder in Fällen, in denen auf sie völlig verzichtet werden könnte, muss während des Annotationsprozesses entschieden werden, ob die fraglichen Passagen möglicherweise in anderen Ebenen als Primärdaten benötigt werden. Auch hierbei ist wiederum zu beachten, dass das System offen ist, d. h. es können neue Annotationsebenen hinzukommen.

---

[50] vgl. Fußnote 20 auf S. 43

[51] Insbesondere gilt dies für den Fall ‚Attributwert oder Elementinhalt'.

Ein weiteres Problem kann in der Sequentialisierung der Primärdaten liegen. Im Allgemeinen kann eine gegebene Abfolge der textuellen Daten angenommen werden, da Textdaten häufig Entsprechungen in einer gedruckten Form besitzen und diese genuin eine Abfolge in sich tragen – vom ersten Buchstaben auf der ersten Seite bis zum letzten Zeichen auf der letzten Seite. Dies steht im Gegensatz zu einer Reihe von Informationen, die üblicherweise in Datenbanken gespeichert werden, die jedoch auch in zunehmenden Ausmaß in der Auszeichnungssprache XML repräsentiert werden (vgl. Abiteboul et al. 2000). Nichtsdestotrotz ergibt sich diese Problematik der Reihenfolgevarianz auch für Textdaten. So werden in Bedienungsanleitungen Hinweise, Warnungen, Produktinformationen o. ä. an variablen Plätzen – vergleichbar mit einer Abbildung – in die Texte eingefügt, z. B. in einem Kasten am Ende einer Seite, wobei der laufende Text um den variabel eingefügten Text herumfließt. Beim Prozess der Textannotation müssen derartige Texte einmal in eine normalisierte Form überführt werden, die bzgl. aller Annotationsebenen invariant ist. Die geschilderte Varianz bei der Textdarstellung bleibt separaten Prozessen vorbehalten, z. B. der Textformatierung oder der Dokumenttransformation mittels geeigneter Sprachen, wie z. B. DSSSL (ISO 10179) oder XSLT (Clark 1999).

Das Sequentialisierungsproblem stellt sich in noch stärkerem Ausmaß für Textdaten, die ihren Ursprung in einem nichttextuellen Medium haben. Bei der Betrachtung linguistischer Daten ist die gesprochene Sprache und deren Transkription von besonderer Relevanz. Von verschiedenen Personen gleichzeitig geäußerte Passagen müssen in eine für alle Annotationsebenen verbindlich festgelegte Abfolge gebracht werden. Die Markierung der partiellen Überlappungen kann, wenn für linguistische Analysen relevant, in einer (separaten) Annotation erfolgen.[52]

Nach der ausführlichen Beschäftigung mit dem Modell und den Voraussetzungen ihrer Anwendung sollen noch kurz unmittelbare Konsequenzen der Verwendung einer derartigen komplexen Annotationsarchitektur erwähnt werden.

- Das Annotationsmodell erlaubt es auf eine natürliche Weise, d. h. ohne ‘workarounds’, Texte zu strukturieren, die nicht als geordnete Hierarchie von Inhaltsobjekten aufgefasst werden können.

[52] Für diese Zwecke bietet sich ein Verweis auf einen Zeitstrahl an, wie sie von der TEI und den Annotationsgraphen verwendet werden. (vgl. Abschnitt 3.2.2)

- Weitere Annotationen können ohne direkte Auswirkung auf die bereits existierenden Ebenen hinzugefügt werden.
- Die Annotationen sind trotz ihrer Unabhängigkeit miteinander eng verknüpft, so dass an jeder beliebigen Stelle der Primärdaten die Informationen aus allen vorhandenen Ebenen der Annotation genutzt werden können.
- Die Kompatibilität einzelner oder aller verwendeter Annotationsebenen kann mittels der Unifikationsoperation (vgl. Abschnitt 4.3.2) überprüft werden.

Neben all diesen aus der Sicht der Informationsmodellierung bedeutsamen Aspekten ist es durch die im letzten Punkte erwähnte Unifikation auch möglich ein wohlgeformtes XML-Dokument für den Datenaustausch bereitzustellen. Wenn die oben definierte Unifikation zweiter Stufe zur Anwendung gebracht wird, ist es immer möglich, ein derartiges Austauschformat zu generieren.

5.2 Wissensrepräsentation annotierter Texte

Annotierte Texte beinhalten im Allgemeinen – wie mehrfach gesehen – Inhaltsdaten und Annotationen. Die Annotationen spielen sich per se auf der syntaktischen Ebene ab, wobei der Zweck der Verwendung dieser syntaktischen Mittel bei der Textaufbereitung darin besteht, (Teil-)Dokumenten Bedeutungen zuzuschreiben. Wenn eine Dokumentgrammatik entwickelt wird, sollte also eine inhaltliche Perspektive ins Zentrum gestellt werden. Diese inhaltliche Sichtweise wird in einer neueren Arbeit von Sperberg-McQueen, Huitfeldt und Renear eingenommen, die zeigt, wie das durch Annotationen rein syntaktisch in die Texte integrierte Wissen extrahiert und mit dem Ziel der Wissensinferenz in den entsprechenden Dokumenten genutzt werden kann (Sperberg-McQueen et al. 2000). Dieser Ansatz wird nachfolgend vorgestellt (Abschnitt 5.2.1). Anschließend wird gezeigt, dass er so verändert bzw. erweitert werden kann, dass er zur Inferenz von Beziehungen primärdatenidentischer SGML- bzw. XML-Dokumente verwendet werden kann (Abschnitt 5.2.2).

5.2.1 Dokumente

Sperberg-McQueen et al. (2000) betrachten annotierte Dokumente als eine Ansammlung von Wissen, welches repräsentiert und inferiert werden kann.

Zum Zwecke der Illustration ihres Ansatzes wird dieses Wissen von ihnen in der logikbasierten Programmiersprache Prolog repräsentiert. Prolog wiederum ist mathematisch abgesichert, in ihr können alle Ausdrücke der Aussagenlogik beschrieben werden. Damit stellt die nachfolgende Benutzung von Prolog keine softwareabhängige Illustration der Theorien dar, sondern wurde schlicht deshalb verwendet, weil die Notation relativ einfach zu lesen ist. Natürlich ist dieser Ansatz, nicht zuletzt wegen des gewählten Illustrationsformats, sehr einfach implementierbar.

Interessanterweise gehört Prolog zu den Programmiersprachen, die für die Computerlinguistik außerordentlich gut geeignet sind und die dort seit langer Zeit Verwendung finden (vgl. Gazdar und Mellish, 1989). Auch wenn Prolog selten im Kontext der Markup-Sprachen Verwendung findet, gab es dennoch bereits Arbeiten, die positive Wechselwirkungen dieser beiden Paradigmen sahen. So hat Bernhard Schröder (1998) ein System vorgestellt, in dem Prolog zum Retrieval von Informationen SGML-annotierter Texte verwendet wurde und in dem sich bereits – wenn auch weniger theoriebezogen – Ideen des Ansatzes von Sperberg-McQueen et al. (2000) finden.

Ein XML-Dokument ohne Querbezüge kann als Baum repräsentiert werden. Als Beispiel dafür soll eine relativ komplexe Repräsentation eines umfangreicheren Textausschnitts dienen, der bezüglich der Dokumentstruktur und der Linguistik annotiert wurde. Der Textausschnitt stammt aus der Bedienungsanleitung eines Toasters[53] und sieht in der gedruckten Form ungefähr folgendermaßen aus:

Reinigen

Sandwichhalter reinigen:

1. Entfernen Sie alle Reste von Brot und geschmolzenem Käse mit einem hölzernen Spatel.
2. Spülen Sie den Sandwichhalter in heißem Wasser, dem Sie etwas Spülmittel zugefügt haben, und trocken Sie ihn gründlich ab. Sie können dieses Teil auch im Geschirrspüler reinigen.

Den Toaster reinigen:

1. Ziehen Sie den Stecker aus der Steckdose.
2. Reinigen Sie die Außenwände des Geräts mit einem feuchten Tuch, auf das Sie bei Bedarf etwas Spülmittel aufgetragen haben. Sorgen Sie dafür, daß kein Wasser in das

[53] Bedienungsanleitung Toaster Philips HD 2572/2574

Gerät eindringt.
Tauchen Sie das Gerät niemals in Wasser.
3. Entfernen Sie Krümel nur, indem Sie die Krümelschublade aus dem Gerät ziehen und leeren.
Schütteln Sie nicht die Krümel aus dem umgekehrt gehaltenen Gerät.
4. Sie können das Netzkabel an der Unterseite aufwickeln, bevor Sie das Gerät zur Aufbewahrung fortstellen.

Die Text besteht aus einer Hauptüberschrift, zwei subordinierten Überschriften und zwei Aufzählungen. Innerhalb der zweiten Aufzählung finden sich zwei verschiedene Formen von Hinweisen. Auf der Ebene der linguistischen Annotation finden sich in den Aufzählungen Imperativ- und Indikativsätze und in den Überschriften Ellipsen bzw. fragmentierte Sätze. Eine Annotation, die zum einen die Strukturierungskonventionen von HTML und das Element `<warning>` verwendet und zum anderen Ellipsen (`<ell>`) und Sätze (`<s>`) annotiert, sieht folgendermaßen aus:

```
<text>
  <h1><ell>Reinigen</ell></h1>
  <h2><ell>Sandwichhalter reinigen:</ell></h2><ol>
  <li><s type='imp'>Entfernen Sie alle Reste von Brot und geschmolzenem Käse
mit einem hölzernen Spatel.</s></li>
  <li><s type='imp'>Spülen Sie den Sandwichhalter in heißem Wasser, dem Sie
etwas Spülmittel zugefügt haben, und trocken Sie ihn gründlich ab.</s>
      <s type='ind'>Sie können dieses Teil auch im Geschirrspüler
reinigen.</s></li></ol>
  <h2><ell>Den Toaster reinigen:</ell></h2><ol>
    <li><s type='imp'>Ziehen Sie den Stecker aus der Steckdose.</s></li>
    <li>
      <s type='imp'>Reinigen Sie die Außenwände des Geräts mit einem
feuchten Tuch, auf das Sie bei Bedarf etwas Spülmittel aufgetragen
haben.</s>
      <s type='imp'>Sorgen Sie dafür, daß kein Wasser in das Gerät
eindringt.</s>
      <warning type='danger of life'>
        <s type='imp'>Tauchen Sie das Gerät niemals in
Wasser.</s></warning></li>
    <li><s type='imp'>Entfernen Sie Krümel nur, indem Sie die
Krümelschublade aus dem Gerät ziehen und leeren.</s>
        <warning type='risk of damage'><s type='imp'>Schütteln Sie nicht die
Krümel aus dem umgekehrt gehaltenen Gerät.</s></warning></li>
    <li><s type='ind'>Sie können das Netzkabel an der Unterseite aufwickeln,
bevor Sie das Gerät zur Aufbewahrung fortstellen.</s></li></ol></text>
```

In einer Repräsentation dieses Textes werden alle Daten- und Elementinhalte als Prolog-Fakten aufgefasst. Würde für den obigen Text eine derartige Repräsentation aufgebaut, sähe der Ausschnitt aus dieser Faktenbasis, der das Wissen über den Satz „Tauchen Sie das Gerät niemals in Wasser“ enthält, folgendermaßen aus:

```
node([1], element(text)).
node([1,5], element(ol)).
node([1,5,1], element(li)).
node([1,5,2,3],element(warning)).
node([1,5,2,3,1],element(s)).
node([1,5,2,3,1,1],pcdata('T')).
node([1,5,2,3,1,2],pcdata('a')).
...
node([1,5,2,3,1,7],pcdata('n')).
node([1,5,2,3,1,8],pcdata(' ')).
node([1,5,2,3,1,9],pcdata('S')).
node([1,5,2,3,1,10],pcdata('i')).
...
node([1,5,2,3,1,39],pcdata('r')).
node([1,5,2,3,1,40],pcdata('.')).
```

Es ist zu sehen, dass ein Prolog-Fakt jeweils einen Knoten des Dokumentbaums enthält. Hierfür wird ein zweistelliges Prädikat `node` (bzw. kurz in Prolog-Notation `node/2`) verwendet, das als erstes Argument eine Liste von Ziffern enthält, die es erlaubt, die Knoten in einer Baumrepräsentation des Dokumentes genau zu lokalisieren.[54] Das zweite Argument ist seinerseits komplex, bestehend aus dem Funktor `element` oder `pcdata` für Element- respektive Dateninhalte und dem Argument, d. h. dem Elementnamen bzw. dem annotierten Datum.

Des Weiteren kommt ein dreistelliges Prädikat `attr/3` als Werkzeug zur Repräsentation von Fakten zum Einsatz:

```
attr([1,5,2,3],type,'danger of life').
attr([1,5,2,3,1],type,'imp').
```

Attribute werden ebenfalls auf Knoten bezogen (Argument 1), sie besitzen einen Namen (Argument 2, hier zufälligerweise zweimal der Attributname `type`) und einen Wert (Argument 3).

Ein derartig repräsentiertes Dokument ist eine außerordentlich gute Basis für das automatische Schließen von Beziehungen, die zwischen den Knoten existieren, wobei hier jedoch nur die im Zusammenhang mit dieser Arbeit relevanten Inferenzen beschrieben werden. So können aus den obigen Fakten automatisch weitere Fakten generiert werden, die das oben enthaltene Wissen in einer einfacheren Form repräsentieren.

---

[54] Diese Lokalisierung ist genauer im Anhang B dargestellt, wo sämtliche Knoten des obigen Beispieltextes aufgeführt sind.

```
...
ol([1,5]).
li([1,5,1]).
warning([1,5,2,3,1]).
...
```

Sperberg-McQueen et al. (2000) erwähnen leider nicht, ob – und wenn ja – wie Dateninhalte in dieser Form repräsentiert werden sollen. Es erscheint allerdings sinnvoll, analog zu der verkürzten Elementrepräsentation, auch für Dateninhalte spezielle Prädikate zu verwenden, wobei jedoch beachtet werden muss, dass in der vereinfachten Notation nicht mehr unmittelbar der Typ des Knotens (`pcdata` oder `element`) erkennbar ist. Da allerdings ein Dateninhalte beschreibender Fakt zwei Argumente besitzt und gleichnamige Prologprädikate mit unterschiedlicher Stelligkeit distinkt sind, kann in der hier verwendeten Prolog-basierten Notation das Prädikat `pcdata/2` Verwendung finden[55].

Die Generierung der Fakten über Dateninhalte resultiert dann in:

```
...
pcdata([1,5,2,3,1,1,7],'n').
pcdata([1,5,2,3,1,1,8],(' ')).
pcdata([1,5,2,3,1,1,9],('S')).
pcdata([1,5,2,3,1,1,10],('i')).
...
```

Diese Notation vereinfacht es insbesondere herauszufinden, welche Fakten für bestimmte Lokationen im Dokumentenbaum Gültigkeit besitzen. Wenn die Fragestellung allerdings andersherum formuliert wird, d. h. wenn gefragt wird, für welche Teile des Baums ein bestimmtes Element gilt, ist es besser, ein weiteres Prädikat zu generieren. Hierfür wird das Prädikat `property_applies/2` eingeführt.

[55] Mit anderen Worten: Es ist dafür gesorgt worden, dass auch ein Element mit dem Namen `pcdata` eindeutig verwendet werden kann, auch wenn nicht mehr – wie in der Ausgangsnotation – erkennbar ist, dass es sich bei `pcdata` um einen Elementbezeichner handelt.

```
...
property_applies(ol,[1,5]).
property_applies(li,[1,5,1]).
property_applies(warning,[1,5,2,3,1]).
...
```

In analoger Weise wird auch das Wissen über Attribute re-repräsentiert.

```
property_applies(type('danger of life'),[1,5,2,3,1]).
property_applies(type('imp'),[1,5,2,3,1,1]).
```

Aus einem in eine solche Faktenwissensbasis überführtes XML-Dokument lassen sich eine Vielzahl von Inferenzen ziehen. Von all diesen potentiellen Schlüssen sind hier nur diejenigen relevant, die jetzt vorgestellt werden.

Für eine gegebene Adresse im Baum kann automatisch inferiert werden, in welche Umgebungen sie eingebettet ist, und für eine gegebene Umgebungsbezeichnung kann erschlossen werden, auf welche Lokationen sich die entsprechende Umgebung bezieht. Für diese Zwecke ist der Term `infer/2` programmiert worden, dessen erstes Argument eine Eigenschaft, z. B. eine Umgebungsbezeichnung, und dessen zweites Argument eine Adresse sein muss. Die Ableitungen werden dann automatisch von dem Prolog-System vorgenommen.

Wird `infer/2` mit einer Adresse aufgerufen, werden alle aktiven Eigenschaften ausgegeben:

```
infer(Eigenschaft,[1,5,2,3,1]).
Eigenschaft = s;
Eigenschaft = warning;
Eigenschaft = li;
...
Eigenschaft = type('imp');
Eigenschaft = type('risk of life');
No.
```

Es ist zu sehen, dass alle im Baum oberhalb dieser Adresse definierten Eigenschaften an die entsprechende Umgebung vererbt werden.[56] Dies erfolgt

[56] Im Text von Sperberg-McQueen et al. werden die damit verbundenen Konsequenzen problematisiert und eine sukzessive Verfeinerung dieses Modells vorgeschlagen. Eines der dort erwähnten Probleme ist in dem obigen Beispiel zu erkennen: die Eigenschaft `type` erhält zwei unterschiedliche Werte, nämlich `'imp'` und `'risk of life'`. Dies ist in diesem Zusammenhang durchaus sinnvoll, in anderen

dadurch, dass die (an der initialen Kapitalisierung erkennbare) Variable 'Eigenschaft' sukzessive alle geltenden Eigenschaften zugewiesen bekommt.[57] Wird `infer/2` hingegen mit einer konkreten Eigenschaft und einer Variablen an der Stelle der Adressenangabe aufgerufen, werden alle Lokationen ausgegeben, für die diese Eigenschaft zutrifft:

```
infer(s,Adresse).
...
Adresse = [1,5,2,3,1];
Adresse = [1,5,3,1];
Adresse = [1,5,3,2,1];
Adresse = [1,5,4,1];
No.
```

In diesem Beispiel werden die Lokationen der Satzknoten ausgegeben (vgl. Anhang B). Obwohl hier nur Aspekte des Ansatzes dargestellt wurden, ist zu sehen, dass der Term `infer/2` ein umfangreiches Mittel der Wissensextraktion zur Verfügung stellt. Zusammenfassend kann festgestellt werden, dass es möglich ist, einzelne annotierte Dokumente als wissensbeinhaltende Einheit – und damit als Wissensbasis – aufzufassen. Dies wurde hier anhand eines spezifischen Ansatzes gezeigt. Ein anderer – jedoch durchaus kompatibler – Ansatz für diese Zwecke wird von Welty und Ide (1999) vorgestellt. Hierbei werden Dokumente in das Wissensrepräsentationssystem CLASSIC eingefügt. Dieser Ansatz beschäftigt sich in noch stärkerem Maße mit den Aspekten der Informationsmodellierung, was sich u. a. daran erkennen lässt, dass die Dokumentgrammatik, d. h. die DTD, als Basis der Modellierung verwendet wird und nicht nur die Dokumentinstanzen. Des Weiteren erlaubt der

---

Fällen, z. B. bei Angaben von Sprachen (z. B. `deutsch` vs. `japanisch`), bestehen jedoch Inkompatibilitäten. Ein anderes Problem betrifft distribuierte und nicht-distribuierte Umgebungen, z. B. gilt die Eigenschaft, dass eine Einheit in einem Manuskript durchgestrichen wurde, für alle seine Teile, d. h. jeden Buchstaben (distribuierte Eigenschaft), wohingegen die Eigenschaft ein Wort zu sein, nur für die gesamte Einheit gilt (nicht-distribuierte Eigenschaft).

Für die vorliegende Arbeit sind diese Verfeinerungen nicht relevant, da es für die hier im Fokus stehende Beschreibung von Beziehungen von Elementen aus unterschiedlichen Dokumenten ausschließlich von Bedeutung ist, dass Elemente in Umgebungen enthalten sind, nicht jedoch die Qualität bzw. Semantik dieser Elemente .

[57] Nach der Eingabe des Semikolons werden weitere Lösungen gesucht. Wenn alle gültigen Lösungen ausgegeben wurden, wird dies vom Prolog-System durch die Ausgabe von `No` vermerkt.

Ansatz von Welty und Ide die Einfügung verschiedener Dokumentinstanzen in die Wissensbasis, wobei diese allerdings derselben DTD genügen müssen.

Bisher ist jedoch nicht gezeigt worden, wie unterschiedliche Dokumente als Basis für Inferenzen verwendbar sind. Um dies zu ermöglichen, soll der in diesem Abschnitt ausführlich vorgestellte Ansatz nachfolgend erweitert werden.

5.2.2 Beziehungen zwischen Dokumenten

Das oben ausführlich dargestellte Wissensrepräsentationsmodell ist auf einzelne Dokumente zugeschnitten worden. Es wird nachfolgend gezeigt, dass dieses Modell so erweitert werden kann, dass es auch zur Inferenz von Beziehungen primärdatenidentischer XML-Dokumente (vgl. Definition S. 65) verwendet werden kann. Zur Demonstration des Ansatzes wird das oben verwendete Beispiel in zwei verschiedene primärdatenidentische Annotationen aufgeteilt.

Wird der Beispieltext ausschließlich nach dem Kriterium der Dokumentstrukturierung annotiert, ergibt sich die Annotation `textstruk`:

```
<text>
  <h1>Reinigen</h1>
  <h2>Sandwichhalter reinigen:</h2>
  <ol>
    <li>Entfernen Sie alle Reste von Brot und geschmolzenem Käse mit einem
hölzernen Spatel.</li>
    <li>Spülen Sie den Sandwichhalter in heißem Wasser, dem Sie etwas
Spülmittel zugefügt haben, und trocken Sie ihn gründlich ab. Sie können
dieses
Teil auch im Geschirrspüler reinigen.</li>
  </ol>
  <h2>Den Toaster reinigen:</h2>
  <ol>
    <li>Ziehen Sie den Stecker aus der Steckdose.</li>
    <li>Reinigen Sie die Außenwände des Geräts mit einem feuchten Tuch, auf
das Sie bei Bedarf etwas Spülmittel aufgetragen haben. Sorgen Sie dafür, daß
kein Wasser in das Gerät eindringt. Tauchen Sie das Gerät niemals in
Wasser.</li>
    <li>Entfernen Sie Krümel nur, indem Sie die Krümelschublade aus dem
Gerät ziehen und leeren. Schütteln Sie nicht die Krümel aus dem umgekehrt
gehaltenen Gerät.</li>
    <li>Sie können das Netzkabel an der Unterseite aufwickeln, bevor Sie
das Gerät zur Aufbewahrung fortstellen.</li>
  </ol></text>
```

Auf eine syntaktische Annotation, die die vollständigen und die fragmentierten Sätze auszeichnet, soll mit `synana` referiert werden:

```
<text>

  <ell>Reinigen</ell>

  <ell>Sandwichhalter reinigen:</ell>

  <s type='imp'>Entfernen Sie alle Reste von Brot und geschmolzenem Käse mit
einem hölzernen Spatel.</s>
  <s type='imp'>Spülen Sie den Sandwichhalter in heißem Wasser, dem Sie
etwas Spülmittel zugefügt haben, und trocken Sie ihn gründlich ab.</s>
  <s type='ind'>Sie können dieses Teil auch im Geschirrspüler reinigen.</s>

  <ell>Den Toaster reinigen:</ell>

  <s type='imp'>Ziehen Sie den Stecker aus der Steckdose.</s>
  <s type='imp'>Reinigen Sie die Außenwände des Geräts mit einem feuchten
Tuch, auf das Sie bei Bedarf etwas Spülmittel aufgetragen haben.</s>
  <s type='imp'>Sorgen Sie dafür, daß kein Wasser in das Gerät
eindringt.</s>
  <s type='imp'>Tauchen Sie das Gerät niemals in Wasser.</s>
  <s type='imp'>Entfernen Sie Krümel nur, indem Sie die Krümelschublade aus
dem Gerät ziehen und leeren.</s>
  <s type='imp'>Schütteln Sie nicht die Krümel aus dem umgekehrt gehaltenen
Gerät.</s>
  <s type='ind'>Sie können das Netzkabel an der Unterseite aufwickeln, bevor
Sie das Gerät zur Aufbewahrung fortstellen.</s>
</text>
```

Unter der Sichtweise der Verwendung separater Annotationen stellt die im Abschnitt 5.2.1 zu betrachtende Gesamtannotation das Ergebnis einer möglichen Unifikation dieser beiden Annotationen und einer wiederum separaten Annotation der Warnungen dar.[58]

Im oben vorgestellten Modell werden für die Überführung der Dokumente in eine Wissensbasis unterschiedliche Baumrepräsentationen angenommen. Wenn z. B. die Repräsentation des Satzes „Sie können dieses Teil auch im Geschirrspüler reinigen.“ entsprechend der beiden Annotationen betrachtet werden soll, ergibt sich bezüglich der Ebene `synana` die folgende Repräsentation:

[58] Auf die Wiedergabe der Annotation der das Element `warning` beinhaltenden Ebene wird hier verzichtet, da zur Illustration der Erweiterung des Repräsentationsformates zwei Ebenen ausreichen.

```
node([1,5,1], element(s)).
attr([1,5,1],type,'ind').
node([1,5,1,1],pcdata('S')).
node([1,5,1,2],pcdata('i')).
node([1,5,1,3],pcdata('e')).
node([1,5,1,4],pcdata(' ')).
node([1,5,1,5],pcdata('k')).
node([1,5,1,6],pcdata('ö')).
...
node([1,5,1,54],pcdata('n')).
node([1,5,1,55],pcdata('.')).
```

Die Baumrepräsentation der Ebene `textstruk` desselben Satzes ist hingegen völlig unterschiedlich:

```
node([1,3,2],element(li)).
node([1,3,2,124],pcdata('S')).
node([1,3,2,125],pcdata('i')).
node([1,3,2,126],pcdata('e')).
node([1,3,2,127],pcdata(' ')).
node([1,3,2,128],pcdata('k')).
node([1,3,2,129],pcdata('ö')).
...
node([1,3,1,177],pcdata('n')).
node([1,3,1,178],pcdata('.')).
```

Um diese unterschiedlichen Repräsentationen nutzen zu können, muss ein absolutes Referenzsystem eingefügt werden. Hierfür sollen die Textteile verwendet werden, die als `PCDATA` gespeichert sind, d. h. eine in beiden Dokumenten identische Zeichenkette dient als absolutes Referenzsystem:

```
Reinigen Sandwichhalter reinigen: Entfernen Sie alle Reste von Brot und
geschmolzenem Käse mit einem hölzernen Spatel. Spülen Sie den Sandwichhalter
in heißem Wasser, dem Sie etwas Spülmittel zugefügt haben, und trocken Sie
ihn gründlich ab. Sie können dieses Teil auch im Geschirrspüler reinigen.
Den Toaster reinigen: Ziehen Sie den Stecker aus der Steckdose. Reinigen Sie
die Außenwände des Geräts mit einem feuchten Tuch, auf das Sie bei Bedarf
etwas
Spülmittel aufgetragen haben. Sorgen Sie dafür, daß kein Wasser in das Gerät
eindringt. Tauchen Sie das Gerät niemals in Wasser. Entfernen Sie Krümel
nur,
indem Sie die Krümelschublade aus dem Gerät ziehen und leeren. Schütteln Sie
nicht die Krümel aus dem umgekehrt gehaltenen Gerät. Sie können das
Netzkabel
an der Unterseite aufwickeln, bevor Sie das Gerät zur Aufbewahrung
fortstellen.
```

Diese Primärdaten bilden eine Zeichenkette mit der Länge 844, woraus sich die folgende Beispielindizierung bezüglich der Rohdaten ergibt (Ausschnitt, vgl. auch Abschnitt 3.7):

```
R e i n i g e n   S a n d w i c h h ...
z1 z2 z3 z4 z5 z6 z7 z8 z9 z10 z11 z12 z13 z14 z15 z16 z17 z18...
...
...
 S   i   e       k   ö   n   n   e   n       d   i   e
z243 z244 z245 z246 z247 z248 z249 z250 z251 z252 z253 z254 z255 z256

...
 ü   l   e   r       r   e   i   n   i   g   e   n   .
z284 z285 z286 z287 z288 z289 z290 z291 z292 z293 z294 z295 z296 z297...
...
...
n   g       f   o   r   t   s   t   e   l   l   e   n   .
z840 z841 z842 z843 z844 z840 z841 z842 z843 z844 z840 z841 z842 z843 z844
```

Eine diese absolute Verknüpfungsbasis beinhaltende, Prolog-basierte Repräsentation der Ebenen `textstruk` und `synana` für den obigen Beispielsatz kann dann entsprechend folgendermaßen aussehen:

Syntaxebene:

```
node(synana,242','297',[1,3,2],element(li)).
node(synana,'242','243',[1,3,2,124],pcdata('S')).
node(synana,'243','244',[1,3,2,125],pcdata('i')).
node(synana,'244','245',[1,3,2,126],pcdata('e')).
node(synana,'245','246',[1,3,2,127],pcdata(' ')).
node(synana,'246','247',[1,3,2,128],pcdata('k')).
...
node(synana,'295','296',[1,3,1,177],pcdata('n')).
node(synana,'296','297',[1,3,1,178],pcdata('.')).
```

Textstruktur:

```
node(textstruk,'242','297' [1,5,1],element(s)).
attr(textstruk,'242','297',[1,5,1],type,'ind').
node(textstruk,'242','243'[[1,5,1,1],pcdata('S')).
node(textstruk,'243','244',[1,5,1,2],pcdata('i')).
node(textstruk,'244','245',[1,5,1,3],pcdata('e')).
node(textstruk,'245','246',[1,5,1,4],pcdata(' ')).
node(textstruk,'246','247',[1,5,1,5],pcdata('k')).
...
node(textstruk,'295',296',[1,5,1,54],pcdata('n')).
node('296',297',[1,5,1,55],pcdata('.')).
```

Es ist zu sehen, dass die Prädikate `node/2` und `attr/3` zugunsten der Prädikate `node/5` und `attr/6` aufgegeben wurden. Die drei zusätzlich eingeführten Argumente werden wie folgt verwendet:

- für die Referenz auf die Annotationsebene (erstes Argument)

- zur Kennzeichnung des Beginns und des Endes (zweites und drittes Argument) der Umgebung in der absoluten Daten- bzw. Referenzebene
- für die Knotenadresse (jetzt Argument 4)
- für die bezeichneten Eigenschaften (Argument 5 bzw. – für Attribute – 5 und 6).

Da alle Informationen der Ausgangsrepräsentation erhalten bleiben, sind weiterhin alle Inferenzen auf der Ebene eines einzelnen Dokumentes möglich, wobei es die Einfügung der zusätzlichen Referenzbasis ermöglicht, annotationsebenenübergreifende Beziehungen zu inferieren. So kann über ein beliebiges Datenelement nicht mehr nur ausgesagt werden, welche Eigenschaften gelten, sondern welche Eigenschaften bezüglich welcher Ebenen der Annotation gelten. Dies ist zweifelsohne auch bei einer vollständigen Separierung der Annotationen in zwei verschiedene Datenbasen möglich, jedoch erlaubt es die Integration in ein System, die durch die Einfügung der absoluten Ebene ermöglicht wurde, jetzt zwischen den Annotationsebenen Beziehungen zu erkennen. Diese Beziehungen bestehen, um exakter zu sein, nicht direkt zwischen den Annotationsebenen selbst, sondern zwischen den in ihnen verwendeten Elementen.

Die möglichen Beziehungen zwischen einem Element e, einer Annotation A – `A:e` – und einem Element e einer Annotation B – `B:e` – können prinzipiell folgende sein[59, 60]:

- Fall (a) Die Umgebung `A:e` umschließt bzw. beinhaltet die Umgebung `B:e` und die Umgebung `B:e` beinhaltet nicht die Umgebung `A:e` (`A:e` $\sqsubseteq$ `B:e` $\wedge$ $\neg$(`A:e` $\sqsupseteq$ `B:e`) bzw. `A:e` $\sqsubset$ `B:e`).

[59] Das Zeichen $\sqsubseteq$ wurde im Abschnitt 4.1.2.2 auch zur Notation der Subsumptionsbeziehung verwendet. Die nachfolgende Verwendung des Zeichens unterscheidet sich von der obigen dahingehend, dass es hier XML-Umgebungen und nicht Attribut-Wert-Strukturen sind, die zueinander in Beziehung gesetzt werden. Gemeinsam ist den Beziehungen, dass sie Relationen von Mengen von Informationseinheiten beschreiben.

[60] Diese prinzipiell möglichen Fälle sind bereits im Abschnitt 4.3.2 ab Seite 105 ausführlich besprochen und exemplifiziert worden. Hier werden jetzt neue Aspekte – Aspekte, die die potentiell automatische Angabe von Beziehungen zwischen den Ebenen betreffen – thematisiert.

- Fall (b) Die Umgebung `A:e` umschließt die Umgebung `B:e` und die Umgebung `B:e` beinhaltet die Umgebung `A:e` (`A:e` ⊑ `B:e` ∧ `A:e` ⊒ `B:e`), d. h. `A:e` und `B:e` umschließen eine identische Primärdateneinheit (`A:e` = `B:e`).
- Fall (c) Es ist nicht der Fall, dass die Umgebung `A:e` die Umgebung `B:e` umschließt und es ist nicht der Fall, dass die Umgebung `B:e` die Umgebung `A:e` beinhaltet.

Es ist zu beachten, dass die beiden durch `e` referierten Elemente trotz desselben Bezeichners distinkt sind, da sie aus unterschiedlichen Annotationen stammen. Formal ergibt sich die Unterscheidung durch die mit Doppelpunkt abgetrennte Voranstellung der Annotationsebenenbenennungen.[61]

Diese Fälle lassen sich auf die im Abschnitt 4.3.2 beschriebenen Unifikationen von XML-Instanzen beziehen. Fall (a) beschreibt die Situation, die von der dort definierten Unifikation erster Stufe abgedeckt ist. Diese Operation ist auch dann erfolgreich, wenn die mit Fall (b) bezeichnete Ausgangsdatenkonfiguration verarbeitet wird. Allerdings gibt es bei der Anwendung der Unifikation erster Stufe auf den Fall (b) kein eindeutiges Ergebnis, eine Situation, die auch nicht durch die Einführung der Unifikation zweiter Stufe ausgeräumt werden konnte. Dass sich die beiden Umgebungen möglicherweise nicht umschließen, sondern alphabetische Varianten zueinander sind oder dass sie unabhängig voneinander denselben Beginn- und Endpunkt besitzen, konnte nicht formuliert werden. Die im Fall (c) beschriebene Situation kann wiederum unterteilt werden: Die Umgebungen besitzen keinen gemeinsamen Dateninhalt – ein unproblematischer (und meist uninteressanter) Fall – oder die Umgebungsgrenzen überlappen sich – die Umgebungen sind inkompatibel. Für den Fall der Überlappung der Elementgrenzen kann festgestellt werden, dass die Unifikation erster Stufe scheitert und nur die Unifikation zweiter Stufe zu einem (bzw. mehreren potentiellen) Ergebnis(sen) führt.

5.3 Hierarchien von Dokumentgrammatiken

Im Abschnitt 5.1 wurde in der Architekturbeschreibung zur Thematik der die XML-Instanzen restringierenden Strukturdefinitionen ausschließlich festgestellt, dass alle zu verwendenden Dokumentgrammatiken referenzierbar sein

[61] Dies geschieht analog zur Verwendung von Namensräumen (vgl. 2.5 Namensräume).

müssen. Die hierfür angenommene Organisationsform wurde als Pool bezeichnet – eine Benennung, die eine Zusammensetzung – eine Nicht-Organisation – zum Ausdruck bringt. Für die Zurverfügungstellung eines innovativen Modells der Annotation textueller Daten ist diese Form der Dokumentgrammatikbereitstellung völlig ausreichend, da das Ziel, Dokumente multipel zu strukturieren, erreicht werden kann. Dennoch sollen elaboriertere Formen der internen Strukturierung dieser Grammatikansammlung aufgezeigt werden.

Die vordergründige Aufgabe von Dokumentgrammatiken besteht darin, Klassen von Dokumenten eine abstrakte Strukturbeschreibung zuzuordnen. Erfolgt diese Strukturierung allein mit dem Ziel der Informationspublizierung, mag eine derartige Sicht ausreichen.[62] Werden Dokumente jedoch mit dem Ziel verschiedenster Auswertungen strukturiert, stellt die Dokumentgrammatik nicht mehr allein eine Dokumenttypendefinition dar, sondern sie bildet vielmehr eine Formalisierung oder sogar Axiomatisierung einer Theorie. Wenn es gelingt, ausgehend von der Betrachtung der Dokumentgrammatiken als einer Implementierung einer Theorie, die Organisation der Dokumentgrammatiken herauszuarbeiten, kann diese Form der formalen Theoriebeschreibung auf eine neue Stufe gestellt werden.

Umgebungen von primärdatenidentischen Dokumenten stehen in Beziehungen zueinander. Dies ist bei der Verwendung nur einer Annotationsebene eine triviale Feststellung, da diese möglichen Beziehungen seit jeher in SGML-basierten Auszeichnungssprachen modelliert werden, nämlich in der DTD. Allerdings sind die in den Dokumenten selbst auftretenden, also wirklich existierenden, Beziehungen bisher selten untersucht worden. Dass dies ein immenser Unterschied sein kann, wird am stärksten durch die Verwendung des Schlüsselwortes `ANY` als Inhaltsmodelldefinition in einer DTD deutlich. Der beschriebene Ansatz von Sperberg-McQueen et al. (2000) ist ein Beispiel für die Modellierung dieser instanziierten Beziehungen auf Dokumentebene. Es wurde durch die Erweiterung und Übertragung dieses Ansatzes auf primärdatenidentische Dokumente mit verschiedenen Dokumentgrammatiken im Abschnitt 5.2.2 erreicht, dass diese Modellierung auch für derartige Dokumente vorliegt. Allerdings wurde bisher noch nicht auf die Möglichkeiten einer abstrakteren, d. h. von den Dokumentinstanzen unabhängigeren Modellie-

[62] Wobei jedoch auch in diesem Fall darauf hingewiesen wurde, dass eine inhaltliche und damit detailliertere Informationsstrukturierung sehr sinnvoll ist. Dies wurde z. B. von Lobin und Witt (1999) am Beispiel von Enzyklopädien gezeigt.

rung, für diese unterschiedlich strukturierten Dokumente eingegangen. Jedoch sollte genau dies ein Ziel wissenschaftlicher Theoriebildung sein.

Bei einer abstrakten Beschreibung der Beziehungen zwischen den Dokumenten müssen die Beziehungen von der Ebene der Instanzen auf die Ebene der Dokumentgrammatiken angehoben werden. Die bisher einzige, existierende (und standardisierte) Möglichkeit derartige Relationen auszudrücken, besteht in der Verwendung von architektonischen Formen (vgl. Abschnitt 3.6), wobei deren Ausdrucksmächtigkeit – wie bereits gesehen – beschränkt ist.

Durch die architektonische Verarbeitung kann einer DTD der Status einer Meta-DTD gegenüber einer anderen DTD zugeteilt werden. Allerdings besitzt dieses Inventarium – wie bei seiner Vorstellung gezeigt – eine eingeschränkte Mächtigkeit. Trotz der Defizite der architektonischen Verarbeitung können architektonische Formen jedoch als gute Ausgangsbasis für eine formale Angabe von Beziehungen zwischen Dokumentgrammatiken angesehen werden. Die im HyTime-Anhang A3: AFDR (ISO 10744, 1997) formalisierten Möglichkeiten stellen die Mindestanforderung an eine abstrakte Beschreibung von Dokumentgrammatikbezügen dar, eine Erweiterung des Instrumentariums ist jedoch unausweichlich.

Mögliche Erweiterungen der vorhandenen Funktionalität der architektonischen Verarbeitung betrifft verschiedene Bereiche:

- Architektonische Beziehungen müssen unabhängig von den Schemasprachen, in denen die Dokumentgrammatik verfasst sind, ausgedrückt werden können.
- Anhäufungen von Elementen umschließende Umgebungen, sollten generiert werden können.[63]
- Elementableitungen sollten in Abhängigkeit von vorhandenen Nachbarelementen vorgenommen werden können, d. h. abhängig vom Auftreten der Elemente in den Dokumentinstanzen sollen unterschiedliche Ableitungen vorgenommen werden können.
- Die Möglichkeiten regulärer Ausdrücke (vgl. Friedl, 1998) sollten in Kombination mit den anderen Erweiterungen Anwendung finden können.

[63] Dies führt zu einer Vergrößerung der Tiefe einer Baumrepräsentation des Dokumentes.

- Umfangreichere Transformationen sollten möglich sein, so dass innerhalb von Umgebungen Verschiebungen von Elementen möglich werden.

Diese Liste dient jedoch ausschließlich einer Skizzierung der potentiellen Erweiterungen. Es ist zur Bildung von Hierarchien zwischen Dokumenten restringierenden Grammatiken nicht notwendig all diese Erweiterungen vorzunehmen. Noch weniger besitzt diese Aufzählung einen Vollständigkeitsanspruch, da es bei einer intensiveren Betrachtung von den in einem Pool von Dokumentschemaspezifikationen angehäuften Strukturbeschreibungen zur Postulierung weiterer Ausdrucksmöglichkeiten kommen könnte.

Nachdem die Beziehungen verschiedener Dokumentgrammatiken untereinander herausgearbeitet wurden, sind sie in einer formalen Notation auszudrücken. Die Notation hierfür muss noch entwickelt werden.[64]

Viele der mit der Thematik der Hierarchien von Dokumentgrammatiken verknüpften Fragestellungen können jedoch zum jetzigen Zeitpunkt nur angerissen werden. Sie sind Gegenstand weiterer gegenwärtiger Forschung.[65]

---

[64] Die in den AFDR verwendeten Notationen wäre ein möglicher Ausgangspunkt, allerdings steht zu vermuten dass sie sich hierfür nicht so gut eignen. Ihre ursprüngliche Version verwendet SGML-Konstrukte, die nicht der XML-Syntax entsprechen. Zwar existieren auch XML-Lösungen, jedoch verwenden diese sogenannte 'processing instructions' als Notation für diese Beziehungen. Processing instructions sollten allerdings – wie der Name bereits besagt – für Verarbeitungshinweise verwendet werden. Die Explizierung von Dokumenthierarchien bildet jedoch ein – von Verarbeitungsprozessen unabhängiges – deklaratives Wissen.

[65] Im Rahmen der DFG-Forschergruppe 437 Texttechnologische Informationsmodellierung (Beginn Herbst 2001) widmet sich das vom Schreiber dieses Textes mitbeantragte Projekt „Sekundäre Informationsstrukturierung und vergleichende Diskursanalyse“ dieser Problematik intensiv.

6 Annotation linguistischer Informationen mit Markup-Sprachen

6.1 Existierende Ansätze

In diesem Kapitel sollen zunächst drei SGML- bzw. XML-basierte Annotationskonventionen vorgestellt werden:

- das DTD Modul der Text Encoding Initiative zur linguistischen Analyse bzw. Interpretation von textuellen Daten (Sperberg-McQueen und Burnard, 1994:455-473)
- der Corpus Encoding Standard (Ide 1998)
- die im europäischen Verbundprojekt MATE entwickelten Möglichkeiten zur morphosyntaktischen Annotation

Im Anschluss daran sollen beispielhaft einige eigene Dokumentgrammatiken beschrieben werden, die es erlauben, linguistisches Wissen zu repräsentieren. Die verwendete Schemasprache ist die im XML-Standard definierte Sprache zur Erstellung von XML-DTDs. Es wurden sehr einfache, jedoch vollständige DTDs entwickelt, die zur separaten Annotation linguistischen Wissens verwendet werden sollen. Aus dieser Anwendungssicht können die DTDs auch als Module aufgefasst werden.

Die Dokumentgrammatiken erlauben die Annotation von linguistischen Ebenen. Diese könnten eine Basis zur später thematisierten Überprüfung (Vgl. Einleitung zum Kapitel 7, S. 159) der Regelwerke kontrollierter Sprachen mittels XML bilden.

6.1.1 TEI

Die Text Encoding Initiative schlägt eine Vielzahl sogenannter Tag-Sets vor, die zur Annotation und Repräsentation unterschiedlicher Textsorten und Theorien verwendet werden können und arbeitet diese in eine modularisierte DTD ein. Einzelne Module können optional verwendet werden, die Verwendung anderer Module ist obligatorisch. Die TEI-DTD definiert durch die obligatorischen Module eine allgemeine Dokumentstruktur in einer mit HTML vergleichbaren Weise. (vgl. Megginson, 1998).

Ein optionales DTD-Modul ist bereits bei der Vorstellung einer SGML-konformen Repräsentation von Merkmalsstrukturen bzw. von Merkmalsstruk-

turbeschreibungen thematisiert worden. (vgl. Abschnitt 4.2, Seite 89). Ein weiteres TEI-Modul kann zur (insbesondere orthographischen) Transkription von gesprochener Sprache verwendet werden. (vgl. Witt et al., 1997; Witt 1998) Ebenfalls mit linguistischen Daten befasst sich das Modul für Wörterbuchdaten, in dem u. a. das bereits oben verwendete Element <syll> definiert wird, welches zur Annotierung von Silben dient.

Ein weiteres – in einem fakultativen Modul definiertes – Tag-Set ist die im folgenden beschriebene Sub-DTD zur Annotation morphosyntaktischer Informationen. Die TEI-Elemente, die zur Annotation derartiger Einheiten verwendet werden können, werden im Kapitel 15 der TEI-Richtlinien vorgestellt. (Sperberg-McQueen und Burnard 1994:475-519) Da einer der generellen Ansprüche der TEI darin bestand, die Annotationen in möglichst geringem Ausmaß theorieabhängig zu spezifizieren, entstand ein sehr allgemeines DTD-Modul, in dem sechs Elemente definiert sind, die es erlauben, linguistische Einheiten zu kategorisieren. Im Einzelnen sind dies:

- Das Element <s> zur Annotation von Sätzen. Mit <s> annotierte Einheiten dürfen nicht ineinander verschachtelt sein. Das hat z. B. zur Folge, dass mit <s> ausgezeichnete Einheiten nicht als die Koordination zweier weiterer <s>-Einheiten repräsentiert sein können. Jedoch steht mit <cl> ein Element zur Verfügung, welches u. a. für diese Zwecke verwendet werden kann.
- Das Element <cl> wird für die Repräsentation von Teilsätzen (engl.: clauses) verwendet. Es erlaubt z. B. die Auszeichnung von Nebensätzen.
- Das Element <phr> dient der Annotation von Phrasen. Diese kommen insbesondere in den Phrasenstrukturgrammatiken zur Anwendung. Eine genauere Spezifizierung der Phrase kann mittels der Attribute type und function vorgenommen werden.
- Mit dem Element <w> werden Wörter, oder genauer Wortformen, ausgezeichnet. Das Attribut lemma erlaubt den Verweis auf das Lexem.
- Das Element <m> erlaubt die Auszeichnung von Morphen. Das diesem Element zugeordnete Attribut baseform kann verwendet werden, um auf eine Repräsentation des Morphems, zu dem das Morph ein Allomorph bildet, zu verweisen. Mit <m> annotierte Einheiten dürfen keine weiteren mit <m> ausgezeichneten Sequenzen beinhalten. Dies hat zur

Folge, dass Zirkumfixe und Infixe nicht in einer intuitiven Form annotiert werden können (vgl. auch Abschnitt 6.1.3).

- Die kleinste annotierbare Einheit ist das Zeichen, welches mit `<c>` repräsentiert wird.

Alternativ zur Verwendung dieser speziellen Elemente besteht auch die Möglichkeit, das sehr allgemeine Element `<seg>` zu verwenden[66] und dieses nach Bedarf zu typisieren. Es wird jedoch in den TEI-Richtlinien darauf hingewiesen, dass dieses Vorgehen die Möglichkeit der Restriktion der Inhaltsmodelle verbaut. An den Stellen, an denen die von der TEI angebotene Kategorisierung zu theorieabhängig erscheint (z. B. ist die grammatische Einheit ‚Phrase' keineswegs unumstritten), kann jedoch auf `<seg>` zurückgegriffen werden.

Als Beispiel einer TEI-konformen partiellen linguistischen Annotation soll der komplexe Satz „Wenn der Toast fertig ist, springt der Schiebeschalter hoch, und das Gerät wird automatisch ausgeschaltet." dienen.[67]

```
<s><cl><cl>Wenn der Toast fertig ist</cl
>, springt der Schiebeschalter hoch</cl
><c>,</c> <w type="conj" function="coord">und</w
> <cl><phr type="np" function="subject"
><w type="det">das</w> <w type="n">Gerät</w></phr
> wird automatisch <m>aus</m><m>ge</m
><m>schalt</m><m>et</m></cl>.</s>
```

Alle Elemente kommen in dieser Beispielannotation zur Anwendung, wobei zum Zweck der besseren Übersichtlichkeit darauf verzichtet wurde, eine konsistente und vollständige Auszeichnung vorzunehmen. Es ist leicht erkennbar, dass der mit diesen Mitteln erreichbare Detailliertheitsgrad der Auszeichnung begrenzt ist. Allerdings ist die TEI-DTD nicht nur modular, sondern auch erweiterbar aufgebaut, so dass sie als Basis für weitere, genauere Annotationen dienen kann. Dieser Weg wird vom Corpus Encoding Standard (CES) beschritten.

---

[66] Das Element `<seg>` wird ebenfalls in einem fakultativen DTD-Modul definiert. Es kann allen TEI-konform annotierten Dokumenten zur Verfügung gestellt werden.

[67] Aus der Bedienungsanleitung: Toaster Philips HD 2572/2574)

6.1.2 CES

Der Corpus Encoding Standard (CES bzw. XCES[68]) ist Teil der Richtlinien der „Expert Advisory Group on Language Engineering Standards“ (EAGLES). CES basiert auf den TEI-Richtlinien, wobei allerdings das soeben beschriebene Modul zur Annotation von geschriebenen Sprachdaten fundamental verändert wird.[69] Die Einbettungsmöglichkeiten in die durch die TEI vorgegebene Dokumentstruktur werden jedoch genutzt, so dass eine komplette Integration in die modulare TEI-DTD möglich wird.

Im CES werden sprachliche Einheiten in der Umgebung `<s>` repräsentiert. Eine oder mehrere dieser Einheiten können ihrerseits gruppiert werden, wofür das Element `<chunk>` zur Verfügung steht. Das Element `<s>` dient, wie in der TEI-Annotation, der Auszeichnung eines Satzes. Im Unterschied zum `<s>`-Element der TEI (`<tei:s>`[70]) darf `<ces:s>` auch verschachtelt sein, d. h. ein `<ces:s>`-Element darf selbst `<ces:s>`-Elemente enthalten. `<ces:s>` bekommt als eine Defaultattributierung `broken='no'` zugeordnet, was ausdrückt, dass ein vollständiger Satz in dieser Umgebung repräsentiert ist. Als Beispiel soll der Hinweis „MEMORY STOP arbeitet nicht bei REC RETURN.“ dienen[71]:

```
<chunkList><chunk><s><tok>
 <orth>MEMORY STOP</orth></tok><tok
><orth>arbeitet</orth><disamb><ctag>verb</ctag
                              ><msd>3PsSg</msd></disamb
               ><lex><base>arbeiten</base
                    ><ctag>verb</ctag></lex></tok><tok
><orth>nicht</orth><ctag>adverb</ctag></tok><tok
><orth>bei</orth></tok><tok
><orth>REC RETURN</orth></tok></s></chunk></chunkList>
```

Es ist zu sehen, dass innerhalb der `<s>`-Umgebung eine sehr detaillierte Annotation der Wörter vorgenommen wurde. Neben der orthographischen Form (`<orth>`) finden sich Angaben über das Lexem der Wortform (`<base>`),

---

[68] XCES bezeichnet die XML-basierte Version des Standards. Die Unterschiede zwischen CES und XCES sind hier irrelevant.

[69] Die fundamentalen Veränderungen kommen faktisch einer vollständigen Neudefinition des Moduls gleich.

[70] Zur Abgrenzung wird gegebenenfalls auf die Namespace-Konvention zurückgegriffen. (vgl. Abschnitt 2.5, S. 38)

[71] Bedienungsanleitung: DENON Stereo Cassette Tape Deck DRM-740

der Wortart (<ctag>) sowie eine genauere morphosyntaktische Beschreibung (<msd>)[72]. Es können alle möglichen syntaktischen Ausprägungen der Wortformen als separate <lex>-Elemente aufgeführt werden. Alternativ hierzu kann, durch Einbettung der wortartenbezogenen und der morphosyntaktischen Informationen in das Element <disamb>, ausgedrückt werden, dass bereits eine Disambiguierung vorgenommen wurde.

An dieser, die wichtigsten CES-Konstrukte zur Annotation einer linguistischen Analyse thematisierenden, Vorstellung ist zu erkennen, dass der Corpus Encoding Standard besonders auf die Annotation der lexikalischen Ebene fokussiert ist. Es gibt kein Äquivalent zu den <tei:phr>-, <tei:cl>- oder <tei:m>-Elementen – ein Manko für eine linguistische Annotation. Der Zweck dieses Standards bestand jedoch darin, eine Möglichkeit der Annotation auf der Ebene der Wörter herzustellen. Während die zu dieser Ebene gehörenden sprachlichen Einheiten innerhalb der TEI-DTD schlicht mit <tei:w> annotiert werden, stellt der CES ein unvergleichbar feingliedrigeres Annotationsinventar zur Verfügung.

6.1.3 MATE

Das bereits in Bezug auf seine Ansätze zur Repräsentation sich überlappender Hierarchien thematisierte Projekt MATE[73] hat das Ziel, Mechanismen und Techniken zur Repräsentation unterschiedlicher linguistischer Beschreibungsebenen zu entwickeln.

Eines der generellen Ziele von MATE bestand darin, bereits existierende Standards zu berücksichtigen. Entsprechend dieses Anspruchs wurden auch die vorgestellten TEI- und CES-Annotationen thematisiert. Beide Standards wurden als vergleichbar mit bzw. auch ähnlich zu den MATE-Entwicklungen bezeichnet, allerdings konnte trotz dieser Ähnlichkeit nicht auf diese bereits existierenden Standards zurückgegriffen werden. Der Corpus Encoding Standard besitzt, wie gesehen, eine zu eingeschränkte Anwendungsmöglichkeit. Aber auch auf das umfangreiche Annotationsformat TEI wurde nicht zurückgegriffen.

Der hierfür genannte Grund ist allerdings nur schwer nachvollziehbar:

---

[72] Diese Informationen sollen in einer in EAGLES definierten Form erfolgen. Hierauf wurde in dem hier gegebenen Beispiel verzichtet. Stattdessen wurde schlicht ‚Verb' bzw. ‚3PsSg' angegeben.

[73] vgl. S. 42 und Abschnitt 3.2.1

> XML is certainly more attractive to work with than is SGML and *hence* TEI.
>
> Dybkjær et al. (1998: 23, Hervorhebung AW)

Zwar ist es zweifelsohne sinnvoll als Annotationsbasis XML zu verwenden, allerdings ist der ‚logische' Schluss keineswegs zwingend. Wie gezeigt wurde (vgl. Witt 1998, DeRose 1999) sind die Unterschiede TEI- und XML-konformer Dokumente auf der Ebene der Dokumentinstanzen sehr gering – sie betreffen insbesondere die leeren Elemente. Eine Kompatibilität zwischen den TEI-Dokumentinstanzen und XML bestand also bereits de facto zur Zeit der Entwicklung des MATE-Annotationsschemas. In jüngerer Zeit sind auch die SGML-basierten DTD-Module der TEI in die Syntax der XML-DTDs überführt worden[74] – ein Prozess bei dem selbstverständlich auch darauf geachtet wurde, dass die Kompatibilität der Dokumentinstanzen erhalten bleibt.

MATE entwickelte Annotationskonventionen und DTDs zur Repräsentation von Dialogstrukturen, von Co-Referenz sowie von Kommunikationsproblemen. Von den ‚klassischen' linguistischen Beschreibungsebenen werden für die Ebene der Prosodie und für die hier besonders relevante Ebene der Morphosyntax Annotationsformate entwickelt.

Das von Pirelli und Soria (2000) beschriebene Modul zur Annotation morphosyntaktischer Informationen verwendet als eine Basisannotation die annotierten Wortformen.[75] Z. B.:[76]

```
<w id="w01">Den</w> <w id="w02">Programmwähler</w
> <w id="w03">nur</w> <w id="w04">bei</w> <w
id="w05">ausgeschalteter</w> <w id="w06">Maschine</w
> <w id="w07">im</w> <w id="w08">Uhrzeigersinn</w
> <w id="w09">drehen</w>
```

Auf diese Basisebene wird von den weiteren Annotationen mittelbar oder unmittelbar Bezug genommen.[77] Für die Annotationsbasis gibt es jedoch trotz

---

[74] Im Juni 2001 wurde unter der Bezeichnung P4 eine XML-Version der modularen TEI-DTD veröffentlicht. (vgl.: www.tei-c.org)

[75] „Tagging at the morphological level presupposes the markup of orthographic words." Pirelli und Soria (2000:66)

[76] Der Beispielsatz „Den Programmwähler nur bei ausgeschalteter Maschine im Uhrzeigersinn drehen." findet sich in der Bedienungsanleitung „AEG Öko_Lavamat 6200, 9200, ..."

ihrer Relevanz keine festgelegte DTD. Die hier wiedergegebene Notation verwendet das auch von der TEI empfohlene Element <w> für orthographische Wortformen. Dieses Element findet sich auch in der Publikation zu MATE, allerdings wird dort für diesen Zweck nicht exklusiv <w> verwendet, sondern auch <word>. Auch wenn eine einheitliche Annotation sicher sinnvoll gewesen wäre, führt die vorhandene inkonsistente Annotationsweise nicht zu negativen Auswirkungen bezüglich der Verbindungsmöglichkeiten mit den anderen Annotationsebenen. Wichtig hierfür ist lediglich, *dass* die Wortformen annotiert und mit Identifikatoren versehen wurden.

In einer separaten Annotation findet die morphologische Auszeichnung statt. Die orthographischen Wörter werden auf ‚morphologische Wörter' (<mw>) oder durch Komposition entstandene ‚morphologische Wörter' (<cpw>) abgebildet. Dies ist meist eine eins-zu-eins Abbildung, wobei die jedoch nicht immer gilt. Z. B. werden klitisierte orthographische Wörter (z. B. regional umgangssprachlich ‚magst's' für ‚magst du es') auf mehrere morphologische Wörter abgebildet. Das Beispiel enthält das Wort ‚im', welches hier auf zwei morphologische Wörter abgebildet ist.[78]

Der reverse Fall tritt z. B. bei Verben mit abtrennbarem Präfix auf (z. B.: tritt ... auf, trennt ... ab). Die mit <mw> oder <cpw> ausgezeichneten Elemente werden anschließend weiter analysiert:

```
<mw  id="mw01" href="#id(w01)"/><cpw id="mw02"
href="#id(w02)"/><mw  id="wm03" href="#id(w03)"><mw
id="mw04" href="#id(w04)"/><mw  id="wm05"
href="#id(w05)"/><mw id="mw06" href="#id(w06)"/>
<mw  id="wm07" href="#id(w07)">in</mw><mw  id="wm08"
href="#id(w07)">dem</mw><cpw id="mw09" href="#id(w08)"/>
<mw id="wm10" href="#id(w09)"
     ><stem>dreh</stem
     ><sufix>en</sufix></mw>
```

Am Beispiel des Wortes ‚drehen' kann gesehen werden, wie eine genauere morphologische Analyse vorgenommen wird. Allerdings ist das zur Verfügung stehende Annotationsinventar nicht sehr elaboriert. So ist es z. B. gemäß der DTD nicht vorgesehen mehr als ein Suffix an einen Wortstamm <stem> an-

---

[77] Dieses Prinzip der Annotation ist bereits im Abschnitt 3.2.1 (ab Seite 46) anhand der Ebene der Prosodie vorgestellt und kritisiert worden.

[78] Dieses Vorgehen dient der beispielhaften Darstellung. Es ist nicht zwingend notwendig – meist sogar nicht sinnvoll – ‚im' mit ‚in dem' gleichzusetzen.

zufügen, was insbesondere für agglutinierende Sprachen, wie z. B. Türkisch oder Finnisch, notwendig ist. Des Weiteren sind die Affixtypen Zirkumfix und Infix nicht annotierbar. Durch diese Einschränkung kann z. B. das im Beispielsatz vorkommende Wort ‚ausgeschalteter' nicht adäquat annotiert werden, da in ihm mehrere Suffixe und das Zirkumfix ‚ge...t' vorkommen.

Die morphologische Annotation, die auf einer Auszeichnung der orthographischen Wörter basiert, ist ihrerseits die Grundlage für weitere Ebenen der Annotation, nämlich für die zwei weiteren Ebenen, für die MATE DTDs zur Verfügung:

- die Ebene ‚chunk' und
- die funktionale Ebene.

Auf der syntaktischen Ebene wird von MATE nicht, wie in der TEI, mit Phrasen und Teilsätzen, sondern mit sogenannten ‚chunks' gearbeitet. Diese Einheiten sind beschrieben als:

> Chunks are textual units of adjacent word tokens which can be linked mutually through unambiguously identified dependency chains with no recourse to idiosyncratic lexical information.
>
> Pirrelli und Soria (2000:80)

Diese Strukturierung führt bezüglich des obigen Beispielsatzes zu der folgenden Segmentierung:

> [Den Programmwähler] [nur] [bei ausgeschalteter Maschine] [(in dem) Uhrzeigersinn] [drehen]

Die Annotation der ersten Einheit verweist dann auf das erste und das zweite morphologische Wort, der zweite ‚chunk' auf das dritte Wort etc.:

```
<ch id="ch01" type="N"    href="#id(mw01)..id(mw02)"/>
<ch id="ch02" type="ADV"  href="#id(mw03)"/>
<ch id="ch03" type="P"    href="#id(mw04)..id(mw06)"/>
<ch id="ch04" type="P"    href="#id(mw07)..id(mw09)"/>
<ch id="ch05" type="FV"   href="#id(mw10)"/>
```

Das hier ohne Inhalt präsentierte Element `<ch>` kann auch für detailliertere Annotationen verwendet werden. In diesen Fällen besitzt es Inhalt, so dass innerhalb der *chunks* Elemente wie `<potgov>` (zur Angabe des potentiellen

dependentiellen Kopfes der Einheit) oder z. B. auch <aux> und <cop> (zur Auszeichnung verschiedener Verbtypen) verwendet werden können.

Die funktionale Struktur eines Satzes kann unabhängig von seiner Zerlegung in *chunks* annotiert werden, da sie nur auf der Auszeichnung der morphologischen Wörter basiert.[79] In dieser Ebene wird die Dependenzstruktur von Sätzen ausgezeichnet. (vgl. z. B. Lobin, 1993:15-36) Die funktionale Struktur des obigen Beispielsatzes, annotiert entsprechend der MATE-DTD, sieht folgendermaßen aus:

```
<funct id="funct_001">
  <head id="h01" href="#id(mw10)"/>
  <dep id="d01" type="dobj" href="#id(mw02)"/>
  <dep id="d02" type="mod" href="#id(mw03)"/>
  <dep id="d03" type="mod" intro="bei" href="#id(mw07)"/>
  <dep id="d04" type="mod" intro="in" href="#id(mw09)"/>
</funct>
```

Der Satz wurde im imperativischen Infinitiv verfasst – ein in Bedienungsanleitungen sehr häufig vorkommender Satztyp. Das Verb ist entsprechend nicht bezüglich Tempus, Person oder Genus Verbi annotiert, was sonst durch Attribute des Elementes <head> erfolgt wäre. Das Verb regiert die vier abhängigen Elemente: das direkte Objekt, zwei durch Präpositionen eingeführte Substantive (intro="bei" bzw. intro="in") und das Adverb ‚nur'.

Aus allgemeineren Veröffentlichungen zu MATE geht hervor, dass es generell möglich ist, weitere Annotationsebenen einzufügen. Das bedeutet, auch wenn von Pirelli und Soria nicht angesprochen, dass zusätzliche Ebenen zur Annotation morphosyntaktischer Ebenen eingefügt werden können. So ist es z. B. möglich in diesem Rahmenwerk eine Phrasenebene oder eine genauere lexikalische Annotation einzufügen.

6.1.4 Diskussion

Die drei vorgestellten Annotionsschemata bilden trotz ihrer Bezugnahme aufeinander sehr unterschiedliche Herangehensweisen an die Auszeichnung von Texten mit linguistischer Information. Die Unterschiede ergeben sich insbesondere in zweierlei Hinsicht: (1) bezüglich der vorgesehenen linguistischen

[79] Jedoch basiert hierdurch auch diese Ebene mittelbar auf der Ebene der orthographischen Wörter.

Beschreibungsebenen und (2) in Bezug auf die Art der Repräsentation der zur Beschreibung verwendeten Daten.

Beschreibungsebenen

Die TEI und MATE reichern die Daten mit Informationen an, die mehrere linguistische Ebenen betreffen. Hierbei sind insbesondere die Morphologie und die Syntax bzw. – im Falle von MATE – unterschiedliche Sichtweisen auf die Syntax zu nennen. CES formalisiert hingegen nur eine Ebene – die Ebene lexikalischer Information. Diese Ebene kann jedoch außerordentlich detailliert annotiert werden.

Das Standardvorgehen der TEI bei der Integration der verschiedenen Ebenen besteht darin, alle Annotationen in dasselbe Datum einzuflechten. Dies kann dazu führen, dass die theoretisch vorhandene Trennung z. B. linguistischer Ebenen verloren geht. Der gravierende Nachteil dieses Vorgehens besteht jedoch darin, dass es prinzipiell vorkommen kann, dass sich Elementgrenzen überlappen und hierfür *workarounds* verwendet werden müssen[80], die einer konsistenten Annotation und damit einer konsistenten Verarbeitung dieser Daten entgegenstehen.

MATE hingegen verteilt die Auszeichnung der verschiedenen linguistischen Beschreibungsebenen in separate Annotationen. Allerdings ergeben sich auch hierbei Nachteile. Die Wahl der Hypertextverknüpfung kann nicht gewährleisten, dass die einzelnen Annotationen auch separat verwendbar sind, da sich durch die Verwendung dieser Techniken notwendigerweise Abhängigkeiten zwischen den Ebenen ergeben. (vgl. auch Abschnitt 3.2.1) Darüber hinaus ist die Annotation nicht nur für Menschen außerordentlich schwer zu lesen und zu interpretieren, sondern macht auch den Einsatz spezialisierter Software erforderlich.[81]

Datenrepräsentation

Die Repräsentation der Inhaltsdaten einer SGML-basierten Dokumentinstanz kann als Primärdaten oder als Attributwerte erfolgen[82]. Es ist bereits darauf hingewiesen worden, dass prinzipiell jeweils beide Wege der Repräsentation

---

[80] Vgl. hierzu insbesondere die Abschnitt 3.3, 3.2.1 und 3.4.

[81] Da eines der Hautziele von MATE darin bestand eine derartige Software zu erstellen und parallel hierzu die Annotationsschemata entwickelt wurden, kann auch spekuliert werden, dass das Annotationsformat auf die Software angepasst wurde.

[82] Die Inhaltsdaten wurden im Abschnitt 2.1.3.1 auf Seite 22 dargestellt, die Bezeichnung Primärdaten erfolgt gemäß der Definition auf Seite 65.

zur Verfügung stehen, sich jedoch dennoch eine Trennung ihrer Verwendungsweise entwickelte. (vgl. auch Fußnote 25 auf Seite 52 und Absatz 5.1) Diese Trennung, aus der auch der hier – in der oben definierten Bedeutung – verwendete Terminus Primärdaten resultiert, besagt, dass die Primärdaten ausschließlich den Beschreibungsgegenstand, d. h. normalerweise die Texte, enthalten und in den Attributen nur Informationen über die Primärdaten enthalten sind.

Werden die Annotationsschemata unter dieser Perspektive betrachtet, lässt sich folgende Einteilung vornehmen:

- Die TEI-basierten Annotationen linguistischer Analysen repräsentieren alle zu beschreibenden Daten als Elementinhalte, d. h. als Primärdaten im obigen Sinn. Es findet keine Duplikation der Daten statt.
- Auch die Annotationen, die vom MATE-Projekt entwickelt wurden, repräsentieren die beschriebenen textuellen Daten als Primärdaten. Informationen über die Primärdaten werden als Attribute oder als – mit den Primärdaten verknüpfte – leere Elemente repräsentiert.
- Der Corpus Encoding Standard nutzt die den Auszeichnungssprachen inhärenten Trennmöglichkeiten nicht, um die konzeptuell vorhandene Wesensverschiedenheit der Informationen auszudrücken. So folgen einem annotierten Wort, z. B. `<orth>`arbeitet`<orth>`, ein Verweis auf das Lexem (`<base>`arbeiten`<base>`), Informationen über die Wortart (`<ctag>`verb`</ctag>`) und die grammatische Kategorie (`<msd >`3PSSg`</msd>`) als Elementinhalt.

Aus der Sicht der hier favorisierten separaten Annotation der verschiedenen linguistischen Ebenen können die folgenden Bewertungen der vorgestellten Annotationsschemata vorgenommen werden.

Die vom CES vorgenommene Beschreibung der Daten ist nicht ohne Modifikationen in eine separate Repräsentation unterschiedlicher linguistischer Beschreibungsebenen integrierbar, da die zur (impliziten) Verknüpfung verwendeten Primärdaten nicht ebenenübergreifend in einer konsistenten Weise zur Verfügung stehen. Die Text Encoding Initiative und MATE vermeiden diese problematische Vermischung der Daten. Der Nachteil der TEI besteht darin, dass die linguistischen Beschreibungsebenen nicht getrennt sind, sondern – wenn immer möglich – in eine Annotation integriert sind. MATE hingegen trennt die Annotationsebenen, besitzt jedoch andere Defizite. Die zur Verfügung gestellten Tag-Sets sind an einigen Stellen außerordentlich theo-

riebezogen, an anderen Stellen nicht so ausgearbeitet, dass sie allgemein verwendbar sind. Da diese Annotationsschemata bis zu einem gewissen Grad austauschbar sind, fällt dieser Punkt jedoch nicht außerordentlich negativ ins Gewicht. Problematisch hingegen ist, dass die Annotationsebenen nur mit großem Aufwand separierbar sind. Dies liegt u. a. darin begründet, dass einzelne Ebenen nicht ohne andere Ebenen verwendet werden können. Hierauf sind auch die DTDs abgestimmt, so dass viele Dokumentgrammatiken existieren, die es den zu ihnen konformen Dokumenten nicht erlauben, Daten (`CDATA` oder `#PCDATA`) zu enthalten. Dies erschwert oder verhindert nicht nur die Verarbeitung mit genereller, d. h. nicht speziell angepasster XML-Software, sondern es widerspricht den Prinzipien der Informationsmodellierung, da auch aus den Dokumentgrammatiken erkennbar sein sollte, an welchen Stellen welche Arten von Daten vorhanden sein können.

6.2 Bausteine zur multiplen Annotation linguistischer Informationen

In diesem Abschnitt werden unterschiedliche DTDs beschrieben, die Komponenten einer linguistischen Annotation bilden, wobei beachtet werden muss, dass sich die Annotation von einer ‚klassischen' Annotation in der im Abschnitt 3.7 und im Kapitel 1 beschriebenen Weise unterscheidet: die textuellen Daten werden mehrfach und separat voneinander annotiert.

Diese hier vorgestellten DTDs erlauben noch keine vollständige oder gar sprachunabhängige Annotation von Texten. Sie besitzen vielmehr Beispielcharakter. Es liegt in der Natur des hier entwickelten Ansatzes, dass die Menge der Dokumentgrammatiken immer unvollständig bleibt, da die Architektur des Annotationssystems keinerlei Festlegungen bezüglich der Dokumentgrammatiken trifft. Dies gilt sowohl in Bezug auf ihre Anzahl und ihre Gegenstandsbereiche als auch für die verwendete Schemasprache.

6.2.1 Silbenebene

Die Annotation der Silben der Wörter resultiert aus der phonologischen Analyse der Wörter. Für textuelle Daten ist die Ebene der Silben insbesondere für die automatische Trennung geschriebener Wörter von Bedeutung. Beim Setzen von Texten dürfen die Wörter an den Silbengrenzen getrennt werden.[83] Aber auch für die automatische Generierung gesprochener Sprache ist die

[83] Dies gilt seit der allgemeinen Einführung der neuen Rechschreibung im Jahr 1998 quasi ausnahmslos.

Silbenstruktur der Texte relevant. Die Dokumentgrammatik zur Annotation der Silben gehört zu den einfachsten hier vorgestellten DTDs.

silbeDTD.dtd

```
<!ELEMENT text (#PCDATA | syll)*>
<!ELEMENT syll (#PCDATA)>
```

Der entsprechend dieser DTD annotierte Textausschnitt „Der Geschäftsführer ...“ sieht folgendermaßen aus:

```
<text><syll>Die</syll> <syll>Ge</syll><syll>schäfts</syll
><syll>füh</syll><syll>rer</syll>...</text>
```

In der einfachen Silbenannotation werden alle Silben ausgezeichnet, die weiteren Textteile werden nicht in MarkUp eingeschlossen.

6.2.2 Morphebene

Die Annotation auf der morphologischen Ebenen kann ausschließlich die konkreten Ausprägungen eines Morphems, d. h. die Morphe bzw. die Allomorphe des Morphems auszeichnen, da das Morphem selbst nur eine abstrakte Einheit ist, die in den Morphen ihre konkrete Ausprägung erfährt. Nichtsdestotrotz werden durch die Annotation der Ausprägung auch die Morpheme selbst annotiert. Die erste Annäherung an die Annotation der Ebene der Morpheme kann in einer zur Silbenannotation analogen Weise erfolgen.

```
<!ELEMENT text (#PCDATA | m )*>
<!ELEMENT m    (#PCDATA)>
```

In einer hierzu vergleichbaren Weise geht die TEI vor. Ein derartiges Annotationsschema ist für die meisten Sprachen ausreichend, da deren Morphologie neben den Wortstämmen nur die Affixtypen Präfix und Suffix kennt. Allerdings gibt es Sprachen, in denen ein Infix in den Wortstamm eingefügt wird, d. h. innerhalb eines Morphs ein weiteres vorkommt oder dass ein Affix – ein Zirkumfix – ein anderes umschließt, wie z. B. bei der deutschen Partizipienbildung. Es muss durch ein allgemeines Basisannotationsschema möglich sein auch diese Morphemtypen zu annotieren. Eine Möglichkeit dem Problem zu begegnen bestünde darin, nicht nur das Element `<m>` für die Annotation der Morphe, sondern auch noch das Element `<mp>` für Teile eines Morphems zu verwenden.

```
<mp morph-type="circumfix" part="1" id="id123"
    affix-ref="id124" >ge</mp
><m morph-type="base">sag</m
><mp part="2" id="id124"
     affix-ref="id123">t</mp>
```

Der Nachteil einer derartigen Lösung besteht jedoch darin, dass verschiedene Elemente zur Annotation von Morphen verwendet werden. Alternativ hierzu kann das Element `<m>` mit Attributen versehen werden, in denen der Typ des Morphems und dessen Vollständigkeitsgrad vermerkt wird. Die alternative Lösung ist weniger restriktiv, als der erstgenannte Vorschlag.

morphDTD.dtd

```
<!ELEMENT text (#PCDATA | m )*>
<!ELEMENT m    (#PCDATA | m )*>
<!ATTLIST m    type (stem|prefix|suffix|
                     infix|circumfix)   #IMPLIED>
```

Ein derartige DTD erlaubt die Annotationen von Morphen, die in Morphen auftreten, z. B. können Infixe in Stämmen auftreten oder ein Zirkumfix kann einen Stamm enthalten (z. B. `<m type='circumfix'>ge<m type='stem' >sag</m >t</m>`). Allerdings erlaubt diese DTD auch die Annotation einer Vielzahl von inakzeptablen morphologischen Analysen. So können nicht nur Wortstämme und Cirkumfixe andere Morphe enthalten sondern auch die verbleibenden Affixe. Dies kann jedoch dadurch verhindert werden, dass die obige DTD als sehr allgemeine Basis-DTD angesehen wird. Zur Annotation können jedoch wesentlich speziellere – sprachspezifische – Dokumentgrammatiken verwendet werden. Z. B. kann die folgende DTD zur Annotation der Morphe einer agglutinierenden Sprache, bei der an einen Wortstamm eine Vielzahl von Suffixe angereiht werden können, verwendet werden.

morphDTDspez-lang1.dtd

```
<!ELEMENT text         (non_morph | (stem, suffix*))*>
<!ELEMENT non_morph    (#PCDATA)>
<!ELEMENT stem         (#PCDATA)>
<!ELEMENT suffix       (#PCDATA)>
```

Diese Dokumentgrammatiken können dann, z. B. mit architektonischen Formen (vgl. Abschnitt 3.6), in eine hierarchische Beziehung zueinander gesetzt werden.

6.2.3 Wortebene

Die einfachste Einteilung textueller Daten in Wörter kann durch die nachfolgende DTD erfolgen, die jedoch noch verändert wird. In ihr wird festgelegt, dass Texte Wörter beinhalten können.

wortDTDprel.dtd

```
<!ELEMENT text (#PCDATA | w )*>
<!ELEMENT w    (#PCDATA)>
```

Auch die TEI verwendet, wie im Abschnitt 6.1.1 zu sehen ist, das Element <w> zur Wortauszeichnung. Im Gegensatz zu dem dort definierten Element <m> erlaubt das Wortelement seine rekursive Einbettung. In den TEI-Richtlinien wird dies dadurch gerechtfertigt, dass z. B. eine Auszeichnung von Wörtern wie (engl.) “isn’t” als <w><w>is</w>n’t</w> möglich sein soll. Das durch die obige DTD definierte Element <w> erlaubt eine derartige Annotation nicht, da dieses Element <w> zur Annotation vollständig auftretender Wortformen verstanden werden soll. Es ist allerdings auch nicht möglich für das in der TEI-DTD definierte Element <w> bzw. <tei:w> eine architektonische Form anzugeben, die es erlaubt, es auf die in wortDTDprel.dtd definierten <w>-Elemente abzubilden. Dies ist nicht möglich, da eine eindeutige Zuordnung der jeweiligen wortannotierenden Elemente nicht hergestellt werden kann. Um <tei:w> verwenden zu können, dürfte seine rekursive Einbettung nicht zur Anwendung kommen.[84] Dies stellt jedoch eine nicht hinnehmbare Einschränkung der Kompatibilität dar. Allein aus diesen Kompatibilitätsgründen wird eine allgemeinere DTD empfohlen.

wortDTD.dtd

```
<!ELEMENT text (#PCDATA | w )*>
<!ELEMENT w    (#PCDATA | w )*>
```

Auch zu dieser DTD kann eine Verfeinerung angegeben werden, die zu wortDTD in einer hierarchischen Beziehung steht. Beispielhaft sei eine derartige DTD angegeben. Sie erlaubt bereits eine detailliertere Auszeichnung.

[84] Interessanterweise unterscheidet sich <tei:w> in dieser Hinsicht sowohl von den schon betrachteten Elementen <tei:m> und <tei:s>, bei denen – wie in Abschnitt 6.1.1 angesprochen – diese rekursive Einbettung wesentlich sinnvoller wäre.

wordDTDclient.dtd

```
<!ELEMENT text (word | non-word | space | punctation)*>
<!ELEMENT word (#PCDATA)>
  <!ATTLIST word  pos  CDATA  #IMPLIED >
<!ELEMENT space      (#PCDATA)>
  <!ATTLIST space xml:space (default|preserve) #IMPLIED>
<!ELEMENT punctation (#PCDATA)>
  <!ATTLIST punctation p-type CDATA  #IMPLIED >
<!ELEMENT non-word (#PCDATA)>
```

Diese speziellere DTD, die vorschreibt, dass alle im Text vorkommenden Einheiten – entweder als Wort, Nicht-Wort (z. B. bestimmte Abkürzungen oder Zahlen), Leer- bzw. Trennzeichen oder als Interpunktion – annotiert wurden, kann in eine architektonische Beziehung zu der obigen allgemeinsten denkbaren, wörterauszeichnenden DTD gesetzt werden. Hierzu kann in einer Architekturdefinition angegeben werden, dass die mit `<word>` annotierten Textteile auf `<w>` abgebildet und die weiteren Annotationen getilgt werden sollen. Die Vorteile der Verwendung einer spezielleren DTD zur Annotation können u. a. darin bestehen, durch Beschränkungen die Texterstellung stärker restringieren zu können oder – wie hier geschehen – das Auftreten von Mixed-Content[85] zu verbieten oder schlicht eine bessere Modellierung sprachlicher Informationen zur Verfügung zu stellen. Immer wenn die Angabe derartiger Architekturen möglich ist, können separate Annotationen automatisch abgeleitet werden. Eine alternative speziellere DTD kann auch so aufgebaut sein, dass noch mehr Elemente verwendet werden, um komplexe Wörter (z. B. Komposita) auszuzeichnen. Auch dies kann durch eine weitere detailliertere DTD erfolgen, die in eine architektonischen Beziehung zu den anderen DTDs gesetzt werden kann.

Für eine Vielzahl linguistischer Anwendungen ist es nicht nur von Bedeutung zu wissen, dass bestimmte Textteile Wörter sind, sondern es ist auch relevant, welchen Wortarten sie zugeordnet werden. Dies kann, wie in wordDTDclient.dtd – der ersten Verfeinerung der DTD – durch die Zuordnung eines Attributes (`type`) oder aber auch – wie in wort-spezDTD.dtd – durch die Definition geeigneter Elemente erfolgen. Eine derartige Modellierung ist in hohem Maße sprach- und theorieabhängig. Eine DTD, die die Wortarten gemäß der Grammatik von Engel beschreibt, kann folgendermaßen aussehen.

[85] vgl. Abschnitt 2.2.3 Mixed-Content auf Seite 33.

wort-spezDTD.dtd

```
<!ELEMENT text
          (Nomen|Pronomen|Verb|Determinativ |
           Adjektiv|Präposition | Subjunktor |
           Konjunktor|Adverb|Modalpartikel|
           Rangierpartikel|Gradpartikel|Kopulapartikel|
           Satzäquivalente|unzuordenbar|
           Leerzeichen | Interpunktion)*>
<!ELEMENT Leerzeichen (#PCDATA)>
<!ELEMENT Interpunktion (#PCDATA)>
<!ELEMENT Nomen (#PCDATA)>
<!ELEMENT Adverb (#PCDATA)>
...
<!ELEMENT unzuordenbar (#PCDATA)>
```

Der aus dem Satz „Die beiden Siebe kontrollieren.“[86] bestehende Text soll beispielhaft in dieser DTD ausgezeichnet werden:

```
<!DOCTYPE text SYSTEM "wort-spezDTD.dtd">
<text><Determinativ>Die</Determinativ><Leerzeichen
> </Leerzeichen><Adjektiv>beiden</Adjektiv><Leerzeichen
> </Leerzeichen><Nomen>Siebe</Nomen><Leerzeichen
> </Leerzeichen><Verb>kontrollieren</Verb
><Interpunktion>.</Interpunktion></text>
```

Eine derartige Annotation auf der Wortebene erlaubt es z. B. mit den Mitteln der DTDs genauere Restriktionen anzugeben (vgl. Lobin 2000), z. B. kann vorgeschrieben werden, dass bestimmte Wortarten, z. B. Verben, in bestimmten Umgebungen, z. B. in Sätzen, vorhanden sein müssen. Die detaillierte sprach- und theorieabhängige Angabe der Wortarten mit der Hilfe von Elementen steht jedoch auch in einer hierarchischen Beziehung zu den oben vorgestellten DTDs. Sie ist spezieller als ‚wordDTDclient.dtd’ und zwangsläufig spezieller als ‚wortDTD.dtd’. Wenn die architektonischen Beziehungen zwischen diesen beiden DTDs definiert wurden, kann aus der angegebenen Beispielinstanz durch eine architektonische Verarbeitung höherer Ordnung automatisch die folgende Instanz abgeleitet werden[87]:

[86] Aus einer Bedienungsanleitung für Constructa-Geschirrspülern.

[87] Eine ausführlichere Darstellung der architektonischen Verarbeitung findet sich im Anhang A.2 (ab Seite 206)

```
<!DOCTYPE text SYSTEM "wortDTD.dtd">
<text><w>Die</w> <w>beiden</w> <w>Siebe</w> <w
>kontrollieren</w>.</text>
```

Es besteht also, wie zu sehen, trotz dieser sehr simplen Basisannotation der Wörter die Möglichkeit, elaboriertere Dokumentgrammatiken anzunehmen und durch standardisierte Methoden automatisch die Ebene der Annotation zu wechseln.

6.2.4 Sätze

Die Auszeichnung der Sätze erfolgt in einer zur Annotation der Morphe analogen Weise. Ein Element `<s>` wird zur Annotation von einfachen und komplexen Sätzen verwendet. Als komplexe Sätze werden Sätze bezeichnet, die aus Haupt- und Nebensätzen bestehen oder die durch Koordination von Sätzen entstanden sind. Es ist zu beachten, dass das Element `<s>` nicht als Platzhalter für das englische Wort *sentence* sondern für das deutsche Wort *Satz* gebraucht wird. Im Englischen wird das Wort ‘sentence’ meist nicht in der hier zu verwendenden allgemeinen Form gebraucht, was erklärt warum das Element `<tei:s>` keine weiteren Elemente `<tei:s>` enthalten darf. Der Begriff ‘clause’ bezeichnet Teilsätze, so dass in der TEI-DTD das Element `<cl>` definiert wurde. Der Nachteil einer Nachbildung dieser Sichtweise in das Annotationsschema besteht darin, dass vollständige Sätze u. U. nicht als Sätze annotiert werden können, z. B. bei der Koordination von zwei Hauptsätzen.

```
<!ELEMENT text (#PCDATA | s )*>
<!ELEMENT s    (#PCDATA | s )*>
<!ATTLIST s    type CDATA   #IMPLIED>
```

Ein entsprechend dieser DTD annotiertes Dokument kann folgendermaßen aussehen:[88]

```
<text><s><s><s>Put the appliance in a safe place</s
>, away from fire hazards such as curtains</s
>, and <s>put the plug in the wall socket</s></s>.</text>
```

Wie schon bei der Wortebene gesehen, kann es u. U. sinnvoll sein, statt eines derartig allgemeinen Annotationsschemas, ausdrucksmächtigere bzw. restriktivere DTDs zu verwenden. Auch für die sehr allgemeine

[88] Operating Instructions: Toaster Philips HD 2572/2574

tivere DTDs zu verwenden. Auch für die sehr allgemeine Dokumentgrammatik ‚satzDTD.dtd' können – in einer analog zu der oben aufgezeigten Weise – speziellere DTDs angegeben werden, die u. U. ‚satzDTD.dtd' als Meta-DTD verwenden können. So können Hauptsätze und Nebensatze, Subordinationen, Koordinationen und verschiedene andere detailliertere Auszeichnungen vorgenommen werden.

Eine weitere, von der Typisierung in Haupt- und Nebensätze unabhängige Kategorisierung der Sätze kann bezüglich ihrer Funktion erfolgen. Die Menge der Sätze kann in Aussage-, Imperativ- und Interrogativsätze unterteilt werden, wobei es weitere, feinere Unterscheidungsmöglichkeiten gibt, z. B. ist der Typ ‚Entscheidungsfragen' eine detailliertere Angabe als die simple Gruppierung als Interrogativsatz.

6.2.5 Phrasen und Chunks

Die vom Annotationsschema MATE vorgesehene Einheit 'chunk' kann ebenfalls durch eine sehr einfache DTD modelliert werden. Auf deren Aufnahme in die hier vorgestellten Komponenten wird jedoch verzichtet, da diese nicht sehr theorieunabhängige Ebene bisher in der Linguistik sehr selten Anwendung findet.

Ähnlich wie die Ebene der 'Chunks' ist die Ebene der Phrasen nicht als theorieunabhängig einzustufen. Allerdings ist das Konzept der Phrase seit einigen Jahrzehnten etabliert und es wird in der linguistische Diskussion relativ schulenübergreifend verwendet. Aus diesem Grund wird eine DTD zur einfachen Annotation von Phrasen zur Verfügung gestellt.

phraseDTD.dtd

```
<!ELEMENT text    (#PCDATA | phr )*>
<!ELEMENT phr     (#PCDATA | phr )*>
<!ATTLIST phr     type CDATA   #IMPLIED>
```

Die Kategorie des Satzes – handelt es sich z. B. um eine Nominalphrase, eine Präpositionalphrase – kann durch das Attribut `type` zum Ausdruck gebracht werden. Es ist allerdings auch hier möglich speziellere DTDs anzugeben, die gegenüber ‚phraseDTD.dtd' – in der Terminologie der architektonischen Formen – den Status einer Client-DTD besitzen. So ist es z. B. möglich – gemäß einer Theorie – alle zu einer Sprache gehörenden Phrasentypen in einer DTD aufzuführen. Es wäre allerdings u. U. manchmal auch

sinnvoll nur einen bestimmten Phrasentyp zu annotieren[89] – ein weiterer Fall für den Einsatz einer DTD, die ‚phraseDTD.dtd' als Meta-DTD verwendet.

6.2.6 Satzfelder im Deutschen

Innerhalb von komplexeren, aus mehreren Wörtern bestehenden, textuellen Einheiten existieren Einschränkungen bezüglich der linearen Abfolge der Komponenten. Diese Wortstellungsrestriktionen sind in hohem Maße von den einzelnen Sprachen abhängig. Im deutschen Nominalkomplex bzw. den Nominalphrasen müssen z. B. das Determinativ vor den Adjektiven und diese vor dem Nomen stehen, im Französischen hingegen werden viele Adjektive den Nomen nachgestellt. Derartige Restriktionen sind durch das Vorhandensein annotierter Wortarten und der Auszeichnung spezieller Phrasen überprüfbar. Auch die Ebene der linearen Abfolge der Phrasen bzw. 'chunks' kann in einigen Sprachen, z. B. im Englischen, durch das Zusammenspiel dieser Ebenen und der Ebene der Satztypen, z. B. Haupt- und Nebensätze und Aussage- und Fragesätze, erfolgen.

Für das Deutsche hingegen ist es notwendig neben der Typisierung der Sätze und der Phrasen eine weitere Ebene zu betrachten, um die Wortstellungsregularitäten adäquat beschreiben und u. U. restringieren zu können. Diese Ebene unterteilt Sätze in die Verbbestandteile, die auf einen linken und einen rechten Teil – die sogenannte Satzklammer – verteilt werden. Um die Satzklammer gruppieren sich die weiteren Satzkomponenten, die als Vorfeld, Mittelfeld und Nachfeld bezeichneten werden. Die Notwendigkeit dieser Einteilung ist in der germanistischen Linguistik unumstritten und findet sich faktisch in allen besseren Grammatiken des Deutschen wieder. Die folgende DTD erlaubt die Auszeichnung dieser Komponenten.

[89] Beispielsweise kann es sinnvoll sein ausschließlich die Nominalphrasen zu annotieren, da nur diese als potentielle Referenten von Pronomen angenommen werden.

satzfelderDTD.dtd

```
<!ELEMENT text (#PCDATA | Vorfeld | left-bracket |
                Mittelfeld | right-bracket | Nachfeld)*>
<!ELEMENT  Vorfeld        (#PCDATA)>
<!ELEMENT  Mittelfeld     (#PCDATA)>
<!ELEMENT  Nachfeld       (#PCDATA)>
<!ELEMENT  left-bracket   (#PCDATA)>
<!ELEMENT  right-bracket  (#PCDATA)>
```

Diese einfache, analog zu den bisherigen Basis-DTDs aufgebaute DTD erlaubt die Auszeichnung der einzelnen Felder. Auch zu dieser DTD soll eine detailliertere Dokumentgrammatik angegeben werden, die jedoch im Gegensatz zu den bisher vorgenommenen Verfeinerungen eine andere Art der stärkeren Restringierung einschlägt. Während ‚satzfelderDTD.dtd' keinerlei Angaben über die Abfolge der Einheiten vornimmt, geschieht dies durch die folgende DTD, die eine potentielle Client-DTD der allgemeinen DTD darstellt.

satzfelderDTDseq.dtd

```
<!ELEMENT text (#PCDATA | s)*>
<!ELEMENT s (Vorfeld?, left-bracket?, Mittelfeld,
             right-bracket?, Nachfeld?)>
<!ELEMENT  Vorfeld        (#PCDATA)>
<!ELEMENT  Mittelfeld     (#PCDATA)>
<!ELEMENT  Nachfeld       (#PCDATA)>
<!ELEMENT  left-bracket   (#PCDATA)>
<!ELEMENT  right-bracket  (#PCDATA)>
```

In Sätzen `<s>` müssen die Felder in der angegebenen Reihenfolge vorkommen. Die Verfeinerungen können jedoch noch viel weitreichender sein, so ist die Optionalität bestimmter Felder abhängig von den Satztypen.

7 Kontrollierte Sprachen

Dieses Kapitel soll darstellen, dass der im Verlauf dieser Arbeit entwickelte Ansatz nicht nur unter einer theoretischen oder konzeptuellen Betrachtungsweise sinnvoll ist. Hierfür wird eine konkrete sprachtechnologische Anwendung vorgestellt: die Kontrollierte Sprache.

Im Abschnitt 7.1 wird zunächst die Materie eingeführt. Hierbei wird – auch unter Bezugnahme auf die historische Entwicklung – gezeigt, dass diese Thematik ständig an Relevanz gewinnt. Darüber hinaus wird ihr wissenschaftlicher Kontext vorgestellt, der jedoch nicht nur erwähnt wird, um das Gebiet Kontrollierte Sprachen möglichst extensiv darzustellen. Vielmehr zeigt seine Beschreibung – inklusive der Vorstellung der Gemeinsamkeiten und der Unterschiede zwischen kontrollierten Sprachen und ihren Verwandten – implizit weitere potentielle Anwendungsgebiete des Ansatzes auf. So befindet sich im Kontext der Kontrollierten Sprachen auch das im Abschnitt 7.1.3 dargestellte Gebiet der Fachsprachen. Es wäre z. B. äußerst interessant, fachsprachliche Texte gemäß des hier entwickelten Ansatzes extensiv annotiert vorzufinden, insbesondere für die stetig an Bedeutung gewinnende Anwendung der Wissensextraktion. Diese potentielle Nutzung der multiplen Annotation wird hier jedoch nicht weiter ausgeführt, sie ist aber implizit vorhanden. Für eine exemplarische Vorstellung des vorhandenen Anwendungsbezuges multipler Annotationen ist es irrelevant, ob neben die sehr wichtigen Anwendung Kontrollierte Sprache weitere Anwendungen gestellt werden.

Im Abschnitt 7.2 werden existierende Regelwerke vorgestellt, wobei einer Kontrollierten Sprache, dem AECMA Simplified English, eine besondere Beachtung zukommt, da dieses Kontrollierte Englisch eine der am häufigsten verwendeten Kontrollierten Sprachen überhaupt bildet.

Bei der Vorstellung der kontrolliertsprachlichen Regeln wird so vorgegangen, dass zum Einen die Regeln selbst wiedergegeben werden, dass zum Anderen jedoch die in den Regelwerken nicht thematisierten Ebenen der Sprachkontrolle aufgezeigt werden. Es wird gezeigt, dass die Sprachkontrolle auf sehr unterschiedlichen linguistischen und nichtlinguistischen Ebenen aufsetzt. Die Herausarbeitung dieser Ebenen ist eine Voraussetzung dafür, Technologien, in deren Zentrum XML steht, direkt zur Sprachkontrolle zu verwenden.

Eine Sprachkontrolle mit den Mitteln von XML könnte am Ende weiterführender Arbeiten stehen. Hierbei fiele den – den XML-basierten Markup-Sprachen

zur Verfügung stehenden – Dokumentgrammatiken (vgl. Absatz 2.4) eine zentrale Rolle zu, da sie es erlauben, die Struktur von Texten zu überprüfen. Gelingt es, die Ebenen der Sprachkontrolle genauer herauszuarbeiten, und werden linguistische und außerlinguistische Phänomene annotiert, so wird es möglich, die Annotation als Basis einer maschinellen Überprüfung der Einhaltung der Regeln zu verwenden. Die inhaltlichen Eigenschaften der Texte werden auf strukturelle Konfigurationen abgebildet. Die Sprachkontrolle ist dann auf der Ebene der Dokumentstruktur angesiedelt. Der Abschnitt 7.2 bietet somit einerseits eine detaillierte Herausarbeitung und Explizierung der Ebenen der Sprachkontrolle, stellt aber andererseits auch eine Grundlage für eine Überprüfung kontrolliertsprachlicher Regelwerke mit den Methoden von MarkUp-Sprachen zur Verfügung.

7.1 Kontrollierte Sprachen: Begriff und Kontext

7.1.1 Ziele und Ideen

Als Kontrollierte Sprachen werden natürliche Sprachen bezeichnet, deren Gebrauch restringiert wird, wobei verschiedene Arten von Restriktion möglich sind. Die Kontrolle kann auf unterschiedlichen linguistischen Ebenen erfolgen und kann der Erlangung unterschiedlicher Ziele dienen. Entsprechend gibt es nicht eine Kontrollierte Sprache, sondern verschiedene. Die den Terminus definierende Gemeinsamkeit aller Kontrollierten Sprachen ist der Ansatz, auf einer existierenden natürlichen Sprache aufzusetzen und diese dergestalt zu variieren, dass bestimmte in der Ausgangssprache korrekte sprachliche Einheiten durch die Sprachkontrolle nicht legitimiert werden. Das bedeutet, dass die Kontrollierte Sprache eine Untermenge der Sprache darstellt, auf der sie aufsetzt. Hierdurch kann jeder, der die Kompetenz in der entsprechenden natürlichen Sprache besitzt, die Kontrollierte Sprache verstehen.

Als das Hauptziel der Sprachkontrolle kann die Vereinfachung der natürlichen Sprache angesehen werden. Eine natürliche Sprache soll so restringiert werden, dass es Menschen leichter fällt, diese Sprache zu rezipieren – ein Vorteil, der insbesondere den Personen, die der entsprechenden natürlichen Sprache nur eingeschränkt gut mächtig sind, dient. Allerdings werden auch Vereinfachungen angestrebt, die eine Verbesserung der Textverständlichkeit zur Folge haben, die allen dient. Die Vermeidung von Ambiguitäten ist hierfür ein Beispiel.

Sprachvereinfachungen haben in jüngerer Zeit auch einen anderen Adressaten – den Computer. Eine maschinelle Interpretation natürlicher Sprache, z. B. zum Zweck ihrer automatischen Übersetzung, ist ein äußerst komplexer Vorgang, der durch das Verbot bestimmter, automatisch nicht (effizient) zu verarbeitender, Äußerungen vereinfacht werden muss.[90] Es ist bisher nicht absehbar, ob es je gelingen kann, natürliche Sprache durch Maschinen so zu verarbeiten, dass den Menschen erlaubt ist, mit Maschinen mittels der menschlichen Kommunikationsform zu interagieren.

Ein weiteres, ebenfalls erst in der jüngeren Vergangenheit relevantes Ziel der Sprachkontrolle besteht darin, den Prozess der kollaborativen Textproduktion zu erleichtern. Insbesondere bei der Erstellung technischer Dokumentationen oder Bedienungsanleitungen kann nicht mehr davon ausgegangenen werden, dass ein Text von einer Person geschrieben wird. Vielmehr setzen sich häufig Gesamttexte aus mehreren Teiltexten zusammen, die von unterschiedlichen Personen erstellt werden. Hinzu kommt, dass die zu dokumentierenden Produkte häufig mehr oder weniger stark modifiziert werden. Dieser Prozess führt zwangsläufig zu einer Modifikation der Bedienungsanleitungen. Die Sprachkontrolle kann dazu beitragen, dass die Rezipienten derartiger Dokumentationen einen Text lesen können, dessen Schreibstil sich nicht gravierend verändert.

Die Kontrollierte Sprache ist ein relativ neues Anwendungsgebiet der Sprach- und Kommunikationswissenschaft – die in diesem Gebiet verwendeten Ziele und Methoden können allerdings mitnichten als neu bezeichnet werden, wie die folgenden Ausführungen zeigen werden.

Natürliche Sprache ist *das* Medium der menschlichen Kommunikation. Kommunikation wird für nahezu alle Menschen erst durch natürliche Sprache ermöglicht. Daraus folgt der seit Jahrhunderten vorhandene Glaube, dass sich Kommunikationsprobleme auf Mankos natürlicher Sprachen zurückführen lassen. Das prominenteste Hindernis für die Verständigung von Menschen mittels Sprachen besteht in der Sprachenvielfalt: der Umstand, dass sich alle Menschen mittels natürlicher Sprache ausdrücken, führt nicht dazu, dass alle Menschen miteinander kommunizieren können. In der Genesis wurde dieses Kommunikationshindernis sogar als eine Strafe Gottes aufgefasst. Allerdings

[90] Es ist keineswegs so, dass Menschen und Maschinen mit denselben sprachlichen Konstruktionen Schwierigkeiten haben. So erkennen z. B. die meisten deutschsprachigen Menschen nicht einmal die Mehrdeutigkeit der Handlungsanweisung „Geben Sie alle Zutaten in einen Topf!“.

stellen die unterschiedlichen Sprachen nicht das einzige Problem sprachlicher Verständigung dar. Auch die Kommunikation innerhalb einzelner Sprachen ist weit davon entfernt komplikationslos zu verlaufen.

Um beiden Problemen entgehen zu können, wurde im Verlauf der vergangenen Jahrhunderte eine schier unüberschaubare Vielzahl von Vorschlägen zu ihrer Lösung unterbreitet. Als Ziele dieser Vorschläge lassen sich insbesondere zwei Lösungsansätze hervorheben:

- Die meisten oder alle Sprachen sind nicht perfekt. Die perfekte Sprache muss gefunden oder entwickelt werden.
- Die betrachtete Einzelsprache oder die menschliche Sprache im Allgemeinen ist zu kompliziert und muss vereinfacht werden.

Beide Ansichten führten zum Teil zu ähnlichen Lösungsvorschlägen, insbesondere weil Kompliziertheit als ein Problem der natürlichen Sprache aufgefasst werden kann, welches diese Sprache zu einer unvollkommenen Sprache werden lässt. Andererseits sind die Bestrebungen der Entdeckung oder Entwicklung von vollkommenen Sprachen nicht notwendigerweise mit der Sprachvereinfachung verbunden.

Umberto Eco beschreibt „Die Suche nach der vollkommenen Sprache“ (1994) in einer sehr detaillierten Form. Die Grundannahme aller dort erwähnten Sprachensuchen liegt darin, dass es eine vollkommene Sprache gibt. Einige Anhänger dieser Idee erklärten ihre eigene Sprache zur vollkommenen Sprache und versuchten, deren Gebrauch zu propagieren. Andere vertraten die Ansicht, dass eine solche Sprache bestimmte Charakteristika besitzt, die Sprachen haben könnten, d. h. prinzipiell kann es eine vollkommene Sprache geben, jedoch existiert diese noch nicht – sie muss erst noch geschaffen werden. Die erste Ansicht entbehrt jeglicher wissenschaftlicher Fundierung und soll daher nicht weiter thematisiert werden, aber auch die These der prinzipiellen Existenz einer vollkommenen Sprache lässt sich allein deshalb nicht aufrechterhalten, da nicht geklärt werden kann, was eine vollkommene Sprache ausmacht. Der Versuch der Entwicklung einer vollkommenen Sprache kann eher als der Versuch einer Entwicklung einer weniger unvollkommenen Sprache betrachtet werden, wobei die Unvollkommenheit daran gemessen werden kann, dass bei der – wie auch immer gearteten – Verwendung der Sprache Probleme auftreten können. Dies bedeutet, dass Sprachen entdeckt oder entwickelt werden müssen, die weniger Probleme verursachen. Das Problem ist: Was ist ein Problem in der Sprache? Als problematisch werden vielfach die Diskurse oder Diskursteile bezeichnet, die die

vielfach die Diskurse oder Diskursteile bezeichnet, die die Kommunikation oder das Erlernen erschweren bzw. sie verkomplizieren.

Bei der Proklamation einer weniger unvollkommenen Sprache lassen sich zwei Herangehensweisen beobachten:

- die apriorische Methode: Eine neue Sprache wird ohne Rückgriff auf existierende Sprachen neu erschaffen.
- die aposteriorische Methode: Eine existierende Sprache wird dergestalt verändert, dass sie keine bzw. nur noch wenige dieser Schwierigkeiten enthält.

Die Kontrollierten Sprachen, die dadurch gekennzeichnet sind, dass sie eine Untermenge einer existierenden Sprache legitimieren, indem die Verwendung bestimmter (komplizierter) Einheiten untersagt wird, sind prototypische Vertreter der Verwendung der aposteriorischen Methode.

Auch wenn Kontrollierte Sprachen, d. h. die regelgeleitete eingeschränkte Verwendung einer existierenden Sprache, erst in jüngerer Zeit Anwendung finden, gibt es in der Geschichte eine umfangreiche Menge von Ansätzen, um einfachere Sprachen zu propagieren.

Die treibende Kraft der historischen Bemühungen der Sprachvereinfachung war meist der Versuch den Menschen ein gemeinsames Kommunikationsmittel zur Verfügung zu stellen. Hierbei sind die Versuche der Einführung von Welthilfssprachen besonders hervorzuheben. Es ist für die Betrachtung der Kontrollierten Sprache sinnvoll, auch diese „entfernten Verwandten“ zu betrachten, da diese Sprachen meist aus dem Bemühen hervorgingen, vereinfachte, sehr regelgeleitete Sprachen zu entwickeln (vgl. auch Lehrndorfer, 1995).

In den Kontext der Kontrollierten Sprachen müssen darüber hinaus auch Fachsprachen eingepasst werden, da sich auch hier einige Parallelen zwischen diesen – bei einer ersten Annäherung – sehr verschiedenen Sprachtypen finden lassen. Die Ähnlichkeiten sollen erwähnt werden, allerdings wird auch eine konkrete Abgrenzung vorgenommen werden.

7.1.2 Kontext: Welthilfssprachen

Zu den bekanntesten apriorischen Sprachen gehören die Welthilfssprachen, auch wenn die Mehrzahl dieser Sprachen als Mischform aposteriorischer und apriorischer angesehen werden kann. Eine Ausnahme bildet die rein apriorische Sprache Solresol, die 1817 von einem französischen Lehrer entwickelt

wurde. (vgl. Bausani 1970) Die Bildung des Lexikons beruht auf der willkürlichen Zuordnung von Tonzeichen, die aus den Musiknoten resultieren. Solresol bildete eine der erste universellen Hilfssprachen[91] und hatte zu ihrer Zeit einen gewissen Erfolg. Insbesondere in der zweiten Hälfte des 19. und am Beginn des 20. Jahrhunderts wurden jedoch eine Vielzahl weiterer Welthilfssprachen entwickelt. In einem strengen Sinne können diese nicht als apriorisch bezeichnet werden, da insbesondere ihre Lexika sprachliche Bausteine verwendeten, die aus existierenden natürlichen Sprachen stammten. Auch Otto Jespersen kreierte eine Welthilfssprache, die er Novial nannte. Im erste Teil der Veröffentlichung von Novial (Jespersen 1928) stellt Jespersen einige der zumindest zu jener Zeit bekannteren Welthilfssprachen zusammen und beschreibt diese prägnant.

Die noch heute bekannteste Welthilfssprache heißt Esperanto. Sie wurde von Ludwig Zamenhof entwickelt und 1887 veröffentlicht. Sie trat – mit einiger Verzögerung nach ihrer Publikation – ab dem Beginn des 20. Jahrhunderts an die Stelle der bis dahin verbreitetsten Welthilfssprache Volapük, die 1870 von Johann Schleyer herausgegeben wurde. Esperanto bildete in vielerlei Hinsicht eine Verbesserung zu Volapük, besaß aber weiterhin schwierige Konstrukte, die dazu führten, dass einige der damals berühmtesten Esperantisten den Esperantodialekt Ido unterstützten (vgl. Jespersen 1928).

Ein Versuch eine Welthilfssprache auf aposteriorischem Wege zu schaffen wurde von Guiseppe Peano beschritten. Er entwickelte 1903 das Latino sine flexione, welches später in Interlingua umbenannt wurde (vgl. Bausani 1970:129ff.). Der ursprüngliche Name beschreibt bereits die Grundideen dieser Hilfssprache: (1) das Lexikon besteht aus lateinischen Wörtern und (2) diese werden ohne Flexion gebraucht. Interlingua besitzt durch sein aposteriorische Bildung viele Parallelitäten zu den Kontrollierten Sprachen, unterscheidet sich aber in einem entscheidenden Punkt von ihnen: Trotz aller Nähe zum Latein ist Interlingua eine künstliche Sprache und keine Untermenge einer existierenden Sprache.

Die Bemühungen der Etablierung einer Welthilfssprache sind in den vergangenen Jahrzehnten allerdings stetig zurückgegangen. Es darf vermutet werden, dass die faktische Etablierung des Englischen dazu führte, dass der

[91] Es gab jedoch auch vor Solresol eine Reihe von Versuchen Welthilfssprachen zu schaffen. Diese können in einem 1885 veröffentlichten Artikel „Zur Geschichte der weltsprachlichen Versuche von Leibnitz bis auf die Gegenwart“ nachgelesen werden. (vgl. Einstein 1885)

Bedarf für ein derartiges Kommunikationsmedium nicht mehr als so hoch eingeschätzt wird.

7.1.3 Kontext: Fachsprachen

Als Fachsprachen werden Varianten der Gesamtsprache bezeichnet, die von Experten für spezielle Formen der Kommunikation, d. h. insbesondere der Wissensübermittlung, verwendet werden. Eine gute Definition von Fachsprachen stellt folglich die am Diskurs beteiligten Personen und die Domäne des Diskurses ins Zentrum. Hoffmann (1998) definiert Fachsprache als „die Gesamtheit aller sprachlichen Mittel, die in einem fachlich begrenzten Kommunikationsbereich verwendet wird, um die Verständigung der in diesem Bereich tätigen Menschen zu gewährleisten“. Beispiele für Fachsprachen sind die Sprachen so differenter Domänen wie der Medizin, der Informatik, der Linguistik, des Schmiedehandwerks oder die Fachsprache der Technischen Dokumentation. Jedoch sind nicht nur die Domänen unterschiedlich, sondern auch die Kommunikationsteilnehmer. Diese Unterschiede lassen es schwierig erscheinen, allgemeine Aussagen über Fachsprachen zu tätigen. Bei genauerer Betrachtung der durch das Bedürfnis nach fachlicher Kommunikation evozierten Diskurse fällt jedoch auf, dass eine Vielzahl von Eigenschaften herausgearbeitet werden kann, die der überwiegenden Mehrheit der Fachsprachen gemein ist. Die Analyse dieser Sprachen wird anhand von Fachtexten vorgenommen, da Fachtexte als eine Manifestation der Fachsprachen angesehen werden (vgl. Baumann 1998). Den natürlich auch auf der Ebene der Fachtexte zu findenden Unterschieden der einzelnen Fachsprachen wird durch die Angabe verschiedener Fachtextsorten begegnet.

Bei einer Beschäftigung mit den vielfältigen Fachtextsorten können als allgemeine Kennzeichen von Fachsprachen markante Eigenschaften auf verschiedenen linguistischen Ebenen herausgearbeitet werden, wobei als Vergleichsbasis sowohl eine nicht näher definierte Standardsprache als auch andere Textsorten, insbesondere auch literarische Texte, herangezogen werden. An erster Stelle unterscheiden sich Fachtexte von anderen Texten auf der Ebene des Wortschatzes. Dieser ist durch die bevorzugte Verwendung einer bestimmten Untermenge von Wortbildungsregeln und durch die Verwendung einer Vielzahl von Fremdwörtern gekennzeichnet. Auch finden sich in Fachtexten – zumindest im Vergleich zur Standardsprache – häufiger Kunstwörter. Aber auch auf der morphologischen, der syntaktischen Ebene und selbst auf der phonologischen Ebene (vgl. Kohrt, 1998b) finden sich fachsprachenspezifische Charakteristika.

Nach der Skizzierung des Begriffs Fachsprachen bleibt zu klären, warum Fachsprachen hier im Kontext Kontrollierter Sprachen eingeordnet werden, insbesondere weil bei einem oberflächlichen Vergleich von Fachsprachen und Kontrollierten Sprachen die Unterschiede wesentlich markanter sind.

Der auffälligste Differenz zwischen Fachsprachen und Kontrollierten Sprachen besteht in der Komplexität: Sprachkontrolle geht nahezu immer mit Sprachvereinfachung einher, wohingegen Fachsprachen außerordentlich kompliziert sind. Der zweite Unterschied ist weitaus weniger auffällig, aus linguistischer Sicht jedoch relevanter: Fachsprachen sind gewachsen, sie erlauben eine diachrone und synchrone Betrachtungsweise. Genau wie die natürliche Standardsprache entwickelt sie sich weiter. Kontrollierte Sprachen hingegen wurden kreiert, eine diachrone Analyse würde ausschließlich auf Modifikationen des Regelwerks schließen lassen, die ohnehin bekannt sind.

Fachsprachen und Kontrollierte Sprachen weisen trotz der genannten Unterschiede eine Reihe von Parallelitäten auf. Beide stellen eine Varietät der Gesamtsprache dar, sie legitimieren jeweils eine Untermenge dieser natürlichen Sprache. Die Menge der in Fachsprachen und Kontrollierten Sprachen vorhandenen Sätze bilden eine Untermenge der Menge der Sätze der Gesamtsprache. Diese Untermengen besitzen ihren Ursprung insbesondere in der verwendeten Terminologie und dem syntaktischen Regelwerk.

Sprachkontrolle erfolgt, wie detailliert an Beispielsprachen im Abschnitt 7.2 zu sehen ist, durch die Einschränkung des natürlichsprachlichen Wortschatzes und der Restriktion der legitimierten grammatischen Strukturen. Dass hierbei Parallelen zu den Fachsprachen zu finden sind, soll an einer kurzen Betrachtung der Ebene des Wortschatzes und der Ebene der fachsprachlichen Syntax demonstriert werden.

Kohrt (1998a) beschreibt aus einer gegenwärtigen Perspektive die auf Fachsprachen bezogene Wortschatzforschung. Die in Fachsprachen verwendeten Wörter wurden lange als die primäre Definitionsebene der Fachsprache angesehen, wobei der Wortschatz sogar bis in die 70er Jahre quasi als die singuläre fachsprachendefinierende Einheit angesehen wurde.

Es wurde bereits erwähnt, dass Fachsprachen gewachsen sind, also von den Benutzern der Fachsprachen in einem kreativen Prozess eingeführt werden. Bei einer genaueren Betrachtung zeigt sich, dass Fachausdrücke nur äußerst selten reine Neuschöpfungen sind. Sie werden vielmehr aus der Menge gemeinsprachlicher Wörter übernommen oder sie werden durch die Komposition oder auch durch Verkürzungen vorhandener Begriffe gebildet. Neben der

kreativen Erweiterung des Fachwortschatzes werden im Anschluss oder parallel zu diesem Prozess Standardisierungen des Fachwortschatzes vorgenommen. Die Fachausdrücke erhalten den Status der Fachterminologie.[92] Die Terminologiearbeit erfolgt von den Fachleuten ihres Gebietes, d. h. nicht von Linguist(inn)en. Die Ziele der Erarbeitung einer Terminologie sind traditionell Klarheit, Exaktheit und Explizitheit.[93] Bei einer expliziten Erarbeitung eines Fachwortschatzes werden die Termini definiert, es wird ihnen eine Kernbedeutung und ein Platz im Terminussystem zugewiesen, wobei auch semantische Beziehungen (Hyperonymie, Hyponymie etc.) vermerkt werden können.

Auch wenn lange Zeit Fachsprachen ausschließlich aus der Perspektive des Lexikons betrachtet wurden, erfolgte nach und nach auch die Ausdehnung der Betrachtungsweisen auf andere linguistische Ebenen, insbesondere auf die Grammatik. Bei der Untersuchung syntaktischer Eigenschaften von Fachsprachen muss beachtet werden, dass keine spezielle Fachsprachensyntax angegeben werden kann. Dies ist insbesondere deshalb unmöglich, da ohnehin nicht von einer Homogenität der Sprachen ausgegangen werden kann. Ziel kann deshalb nur sein, bestimmte Eigenschaften auf der syntaktischen Ebene herauszuarbeiten, die den Fachsprachen gemein sind. Hierbei zeigt sich allerdings, dass es eine allgemeine Formulierung der syntaktischen Eigenschaften erlaubt, die deutliche Mehrheit der Fachsprachen zu charakterisieren und somit syntaktische Eigenschaften von Fachsprachen herauszuarbeiten, die nicht nur einzelsprachlich Gültigkeit besitzen. Hierbei besteht auf der anderen Seite allerdings das Problem, dass zu allgemeine Formulierungen gefundener Regularitäten nicht formalisiert werden können, was insbesondere den Einsatz von Computern bei der Theoriebildung erschwert.

Hoffmann (1998) fasst die syntaktischen Eigenschaften von Fachsprachen zusammen, und zwar aus verschiedenen Blickwinkeln, die nicht per se auf die Ebene der klassischen grammatischen Analyse zurückzuführen sind. Bereits das erste von ihm betrachtete Kriterium, die Satzlänge, ist ein Kriterium, welches sich nicht direkt der Syntax zurechnen lässt. Die anderen von ihm betrachteten Eigenschaften von Fachsprachen betreffen die verwendeten Satz-

---

[92] Nach Göpferich (1998) wird der Begriff Terminologie mehrdeutig gebraucht. Neben der hier verwendeten Bedeutung als standardisierte Fachausdrücke, die z. B. in *Terminologie*datenbanken gespeichert werden, wird der Begriff auch allgemein – quasi synonym zu Fachwortschatz – verwendet.

[93] Dies gilt nach wie vor, auch wenn in jüngerer Zeit diese Ziele kritisch hinterfragt werden. Kohrt (1998a) merkt an, dass diese Ziele den Wert von sprachlicher Unschärfe für die Kommunikation nicht anerkennen.

arten, die Satzeigenschaften, die Satzgliedfolge unter Berücksichtigung der Gliederung bezüglich Thema und Rhema, der Valenzbeziehungen der Konstituenten, der Kompression und den Mitteln der Anonymisierung. Es zeigt sich, dass diese Analyseebenen einerseits Regularitäten erkennen lassen, die Fachsprachen von anderen Subsprachen abgrenzen, andererseits aber ebenfalls gute Indikatoren für unterschiedliche Fachsprachen darstellen. Wie angedeutet lassen sich nicht alle dieser Kriterien im strengen Sinn auf die Syntax zurückführen. Dies zeigt sich unter anderem auch an der Betrachtung der Redundanzarmut – einem sehr charakteristischen Kennzeichen von Fachtexten.

Redundanzarmut wird durch Kompression des Inhalts bzw. durch Informationsverdichtung erreicht. Hierfür werden lexikalische und typographische Mittel verwendet. Durch die Einbindung nichtverbaler Informationen werden darüber hinaus auch außersprachliche Möglichkeiten verwendet. Von besonderer Bedeutung sind jedoch die syntaktischen Möglichkeiten, da diese in besonderem Maße an der Konstituierung einer eigenen Textgattung „Fachtext“ beteiligt sind:

> Neben der Reduzierung von Nebensätzen auf *Partizipial-* und *Gerundialkonstruktionen* (im Russischen Adverbialpartizipien) werden *Genitiverweiterungen, präpositionale Substantivgruppen, einfache und erweiterte Attribute, Partizipialgruppen, Ellipsen, Aufzählungen* und die *Asyndese* als Kondensationsformen erwähnt, die für Fachtexte typisch sind.
>
> Hoffmann (1998:421)

Ein weiteres Charakteristikum von Fachtexten bildet der hohe Grad der Anonymisierung. Auch hierfür stehen verschiedene Wege zur Verfügung, jedoch bietet die Ebene der Grammatik das umfangreichste Repertoire, genannt seien hier nur die Verwendung eines Nominalstils (komplexer Nominalgruppen, Deverbalisierung) und die häufige Verwendung von Passivkonstruktionen.

Zusammenfassend lässt sich festhalten, dass die Syntax von Fachsprachen durch Informationskompression und durch Anonymisierung geprägt wird. Eine derartige Sprachverwendung steht allerdings im Widerspruch zum Ziel der Verbesserung der Verständlichkeit. Aus der Perspektive der Kontrollierten Sprachen betrachtet, bedeutet dies, dass die sich innerhalb der Fachsprachen herausgebildeten syntaktischen Mittel keineswegs in das Regelwerk übernommen werden sollten, wenn die Sprachkontrolle (auch) der Erhöhung des Verständlichkeitsgrades dienen soll. Die Erfolge der Terminologiearbeit,

die zu elaborierten fachgebietsbezogenen Terminologiesammlungen geführt haben, stellen jedoch eine äußerst gut geeignete Basis für die Sprachkontrolle auf der Ebene des Lexikons dar.

7.2 Regelwerke für Kontrollierte Sprachen

7.2.1 Überblick

In den folgenden beiden Abschnitten werden verschiedene Kontrollierte Sprachen – ein vollständig definiertes kontrolliertes Englisch und Grundzüge eines kontrollierten Deutsch – vorgestellt werden. Diese werden vorher durch einen kurzen historischen Überblick und die Erwähnung typischer Vertreter bestimmter Sprachkontrollen in einen Kontext gesetzt werden.

Die älteste restringierte Sprache – ein Vorläufer der Kontrollierten Sprachen – ist Charles K. Ogdens 1932 publiziertes BASIC-English, bei dem die Sprachkontrolle insbesondere auf der Ebene des Lexikons stattfand. Sie sollte der Verbesserung der Kommunikation mit den Bewohnern der britischen Kolonien dienen. Es wurde eine Liste von 850 Wortformen definiert, die jedoch ambig verwendet werden durften, z. B. sowohl als Verb als auch als Substantiv. Es gab auch Bemühungen diese Restriktionen auf das Deutsche und das Französische zu übertragen. Vergleichbar mit diesen Sprachen war das in den 60er Jahren entwickelte Caterpillar Fundamental English (CFE), welches dazu dienen sollte, Technische Dokumentationen zu verfassen, die nicht mehr in andere Sprachen übersetzt werden mussten, da die verwendete Sprache so einfach ist, dass sie äußerst schnell von den die Dokumentationen lesenden Technikern erlernt werden kann. (vgl. Huijsen 1998)

In jüngerer Zeit wurde eine Vielzahl Kontrollierter Sprachen entwickelt, die im Gegensatz zu ihren Vorläufern die Restriktionen nicht allein auf der Ebene des Lexikons, sondern auch auf anderen linguistischen Ebenen vornehmen. Huijsen (1998) hat eine Vielzahl dieser Sprachen zusammengestellt, Sprachen die meist von Firmen zur Verbesserung ihrer technischen Dokumentationen entwickelt wurden. So entwickelte der Nutzfahrzeughersteller Scania das ScaniaSwedish, eine Kontrollierte Sprache, die die automatische Übersetzung vereinfachen sollte. Mit demselben Ziel entwickelte Siemens das Siemens Dokumentationsdeutsch (vgl. Abschnitt 7.2.3)).

Im Gegensatz zu diesen firmeneigenen, sehr spezialisierten Entwicklungen, die mithin firmeninternes Wissen darstellen, wurde das AECMA Simplified English, (SE, vgl. Abschnitt 7.2.2) von einem Konsortium, einem Zusammen-

schluss von Firmen aus der europäischen Luft- und Raumfahrtindustrie, entwickelt. Wie bei allen Kontrollierten Sprachen ist das Ziel der Kontrolle nicht nur die Spezifizierung eine vereinfachten, ambiguitätsärmeren und dadurch exakteren Kontrollierten Sprache, sondern auch eine domänenspezifische – wenn auch nicht firmenspezifische – Sprachrestriktion. Das AECMA SE wurde neben seiner konkreten Anwendung in der technischren Dokumentation auch als Basis für die Entwicklung eines Kontrollierten Französisch mit dem Namen GIFAS français rationalisé bzw. GIFAS Rationalised French (Barthe et al. 1999) verwendet.

Bei der Betrachtung Kontrollierter Sprachen fällt auf, dass die Entwicklung hauptsächlich im außeruniversitären Rahmen verlief. Allerdings sind auch im akademischen Umfeld Kontrollierte Sprachen entwickelt worden. Rolf Schwitter (1998) stellte in seiner Dissertation mit Attempto Controlled Language (ACE) ein kontrolliertes Englisch vor. In ACE formulierte Äußerungen können eindeutig in eine semantische Repräsentation in der Diskursrepräsentationstheorie (DRT, vgl. Kamp und Reyle 1993) überführt werden. Diese semantische Modellierung erlaubt die Interaktion mit einem Auskunftssystem in einer (kontrollierten) natürlichen Sprache.

Ebenfalls in einer Dissertation wurden Regeln für ein Kontrolliertes Deutsch entwickelt, die auch sprachpsychologische Aspekte berücksichtigen. (Lehrndorfer 1996) Diese Regeln werden bei der Diskussion der Thematik Kontrolliertes Deutsch (Abschnitt 7.2.3) thematisiert.

7.2.2 AECMA Simplified English

Das AECMA Simplified English besteht aus zwei Teilen: dem Regelwerk und dem Lexikon. Das AECMA SE gehört somit zu denjenigen Kontrollierten Sprachen, die ein explizites Lexikon verwenden. Dieses Lexikon ist nicht fix, d. h. weitere Lexeme können verwendet werden, wobei diese jedoch nur bedingt frei gewählt werden können. Es existieren vielmehr im Regelwerk Anweisungen, die die Verwendung dieser neuen Lexeme restringieren.

Die nachfolgenden Abschnitte beschreiben AECMA SE unter besonderer Berücksichtigung der linguistischen Ebenen, die von der Sprachkontrolle betroffen sind. Dies hat zur Folge, dass die Organisation des Regelwerks verändert wurde. Die Darstellung hier beginnt mit der Beschreibung des zweiten Teil der AECMA-Richtlinien – dem Lexikon. Anschließend werden die wortformen- und lemmataauswählenden Regeln, die syntaktischen und die gestaltungsspezifischen Regeln, beschrieben. Die originale Gliederung ist durch die

Beibehaltung der ursprünglichen Nummerierung erschließbar. Die ursprüngliche Einteilung des Regelwerks erfolgte nach dem folgenden Schema: (1) Wörter, (2) Nominalphrasen, (3) Verben, (4) Sätze, (5) Prozeduren (Handlungsanweisungen), (6) Beschreibungen, (7) Warnungen, (8) Interpunktion und (9) Schreibpraxis.

Die Zuordnung der Regeln zu diesen Kategorien kann aus der ersten Zahl der Nummerierung ersehen werden.

7.2.2.1 Das Lexikon

Die Wortliste

Im AECMA Lexikon sind die zugelassenen Wörter tabellarisch explizit aufgelistet. Darüber hinaus sind untersagte Wörter und erlaubte Alternativen für diese Wörter aufgeführt. In der folgenden Tabelle ist ein Ausschnitt aus dem Lexikon zu sehen.

INCIDENT (n)	An important „occurrence“ that can cause damage or have dangerous results	RECORD ALL INCIDENTS OF WATER FOUND IN THE FUEL.	
incline (n)	SLOPE	IF YOU MUST TOW THE AIRCRAFT DOWN A SLOPE, THERE MUST BE A PERSON IN THE COCKPIT TO OPERATE THE BRAKE IF NECESSARY.	If the aircraft has to be towed down an incline, there must be someone in the cockpit to operate the brake if necessary.
INCLUDE (v), INCLUDES, INCLUDED, INCLUDED	To make, or to be, part of	THIS CHAPTER INCLUDES THE PROCEDURES FOR THE REMOVAL OF THE LANDING GEAR.	
Including (pre)	THRU	DO TESTS 4 THRU 8 AGAIN.	Repeat from test 4 up to and including test 8.
	WITH	RETURN THE DEFECTIVE COVER, WITH THE OIL SAMPLES, TO THE REPAIR CENTER.	Return defective cover, including oil samples, to the repair center.

Tabelle 1: : Ausschnitt aus dem AECMA Simplified English Lexikon

Die erste Spalte enthält das spezifizierte Lemma. Die Lemmata sind in Minuskeln geschrieben, wenn sie nicht verwendet werden sollen, in Majuskeln,

wenn sie erlaubt sind. In Klammern steht in beiden Fällen die Spezifikation der Wortart.

Für die erlaubten Lemmata sind die zulässigen Wortformen in Großbuchstaben aufgeführt. Die Tabellenzeile enthält für diese Lexeme in der zweiten Spalte eine Spezifikation der Semantik und in der dritten Spalte eine beispielhafte Verwendung einer Wortform. Die vierte Spalte enthält keine weiteren Informationen. Die genaueren Richtlinien zur erlaubten und untersagten Terminologie sind in weiteren Regeln festgelegt.

Für die nicht zu verwendenden Wörter finden sich in der zweiten Spalte die in Großbuchstaben geschriebenen empfohlenen Synonyme. Die vierte Spalte enthält einen Beispielsatz für die Verwendung des inakzeptablen Wortes und die dritte Spalte eine Paraphrasierung gemäß den Richtlinien des AECMA Simplified English.

Im Zusammenhang mit der hier vorgelegten Arbeit wurde eine XML-annotierte Version des vom AECMA-Lexikon erstellt. Der in der Tabelle 1 zu findende Ausschnitt des Lexikons sowie die entwickelte DTD sind im „Anhang C: Kontrollierte Sprache“ (ab Seite 211) dargestellt.

Der Vorteil der entwickelten XML-Version besteht insbesondere darin, dass die in der Tabellenform implizit vorhandenen Informationen expliziert wurden. So wurde die durch die Verwendung von Majuskeln und Minuskeln angezeigte Information über die Zulässigkeit der enthaltenen Wortformen direkt durch die Einführung der Elemente `<approved>` und `<non-approved>` gekennzeichnet. Die weiteren Konkretisierungen betreffen u. a. auch das grammatisches Wissen. So werden z. B. die – in der Tabellendarstellung durch die Reihenfolge der Aufführung der Wortformen ersichtlichen – grammatischen Formen der Verben durch XML-Attribute expliziert.

Des Weiteren wurde eine kompaktere, nur die zulässigen Wörter beinhaltende XML-Version der erstellt:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Lexicon [
<!ELEMENT Lexicon (word+)>
<!ELEMENT word         (#PCDATA)>
<!ATTLIST word     id          ID          #IMPLIED
                   pos         CDATA       #REQUIRED
                   lexem       (yes|no)    #REQUIRED
                   form        CDATA       #IMPLIED
                   reference   IDREF       #IMPLIED>]>
<Lexicon> ...
  <word id="incident_n1" pos="noun"
        lexem="yes">incident</word>
  <word id="include_v1" pos="verb"
        lexem="yes">include</word>
  <word pos="verb" lexem="no" form="vform1"
        reference="include_v1">includes</word>
  <word pos="verb" lexem="no" form="vform2"
        reference="include_v1">included</word>
  <word pos="verb" lexem="no" form="vform3"
        reference="include_v1">included</word>
...</Lexicon>
```

Alle durch das AECMA-Lexikon legitimierten Wortformen sind in dieser Version enthalten. Jede Wortform ist als textueller Inhalt im Element `<word>` enthalten, die grammatischen Informationen sowie Verweise auf Haupteinträge sind in den Attributen enthalten.

Lexikalische Regeln

Es liegt in der Natur der lexikalischen Regeln, dass sie die Ebene des Lexikons betreffen. Für eine Überprüfung der korrekten Verwendung des Lexikons mit texttechnologischen Mitteln ist eine Annotation auf multiplen – insbesondere linguistischen – Beschreibungsebenen nicht zwangsläufig notwendig.

Das Lexikon

Die Verwendung der im Lexikon in Großbuchstaben aufgeführten Wortformen unterliegt folgenden Regeln:

Regel 1.2: Die im Lexikon enthaltenen Wortformen dürfen nur in der angegebenen Wortart verwendet werden.

Durch diese Regel wird u. a. eine sehr produktive Sprachverwendung des Englischen ausgeschlossen, die in der humoristischen Aufforderung „Let's verb nouns" zum Ausdruck gebracht wird.

Regel 1.3 Die im Lexikon enthaltenen Wörter dürfen nur in der angegebenen Bedeutung verwendet werden.

Hierin wird eine Ausweitung der auf die syntaktische Ebene bezogenen Regel 1.2 auf die Semantik vorgenommen. Beide Regeln gemeinsam besagen, dass alle Wörter ausschließlich in der angegebenen, festgelegten Weise verwendet werden dürfen. Homographen, die eine andere Bedeutung oder Wortart besitzen, dürfen – falls nicht explizit aufgeführt – nicht verwendet werden.

Regel 1.4 Die im Lexikon enthaltenen Verben und Adjektive dürfen nur in den explizit angegebenen grammatischen Formen verwendet werden.

Diese Regel wird in Regel 3.1 mit der die Diskussion der zulässigen Tempora eingeführt wird partiell wiederholt.

Regel 3.1 Es sind nur die Verbformen zu verwenden, die explizit im Lexikon aufgeführt sind.

Die Regeln 1.4 und 3.1 drücken aus, dass für Verben und Adjektive alle verwendbaren Wortformen im Lexikon aufgeführt sind. Implizit wird allerdings durch die Festlegung in der Regel 1.4 auf Verben und Adjektive zum Ausdruck gebracht, dass die Wörter anderer Wortarten durchaus in anderen Wortformen verwendet werden dürfen. Für Funktionswörter hat dies keine Konsequenzen, da ihre Wortform mit dem Lexem übereinstimmt. Die neben den Adjektiven und Verben existierende Gruppe von Inhaltswörtern, die Nomen, können jedoch in unterschiedlichen Wortformen verwendet werden. Von besonderer Bedeutung sind hierbei die Pluralformen, deren Verwendung durch Regel 1.4 somit implizit ermöglicht wurde.

Trotz dieser Rahmenbedingungen ist der Wortschatz, die Menge der legitimierten Lexeme, erweiterbar. Dies wird bereits in der ersten Regel verdeutlicht:

Regel 1.1: Es dürfen nur die im Lexikon aufgenommenen Wörter sowie Fachterminologie verwendet werden.

Eine genauere Spezifikation der Fachterminologie erfolgt in den weiteren Regeln. Es werden in separaten Regeln zwei Kategorien von Fachterminologie, (1) Technische Ausdrücke (technical names) und (2) Ausdrücke, die Fertigungsverfahren bezeichnen (Manufacturing Processes) eingeführt.

Technische Bezeichner

Regel 1.5 Technische Benennungen (Technical Names) dürfen verwendet werden, auch wenn sie nicht im Lexikon enthalten sind.

Technische Benennungen sind nicht in das Lexikon aufgenommen worden. Im Regelwerk zum SE sind unter der Regel 1.5 zwanzig Kategorien aufgeführt, die es erlauben technische Namen zu erkennen und zu gruppieren. Hierzu gehören u. a. Bezeichnungen für Werkzeuge, (Flugzeug-)Teile, mathematische Einheiten, Zahlen oder für persönlichen Besitz.

Regel 1.6 Technische Benennungen müssen als Substantiv oder als Adjektiv gebraucht werden.

In dieser Regel und der zugehörigen Ausführung wird keine präferierte Verwendungsweise angegeben, jedoch wird aus der Auswahl der Beispiele deutlich, dass die Verwendung technischer Bezeichner als Adjektiv den Ausnahmefall darstellt.

Regel 1.7 Eine technische Benennung muss in der offiziellen Terminologie oder standardsprachlich erfolgen.

Insbesondere für Fachsprachen ist es üblich Terminologien genau festzulegen. Diese sollten zur Anwendung kommen, wobei jedoch auf spezielle, relativ unbekannte Bezeichnungen verzichtet werden sollte. Sollten keine Terminologien standardisiert worden sein, sollte beim Verfassen kontrolliertsprachlicher Texte ebenfalls auf eine zu spezielle Terminologie verzichtet werden, wobei allerdings auch keine umgangssprachlichen Bezeichnungen (slang) in den Texten vorkommen sollten.

Regel 1.8 Für dieselben Konzepte sollen keine unterschiedlichen Bezeichnungen verwendet werden.

Einmal eingeführte technische Bezeichner sollten konsistent weiterverwendet werden. Diese Regel steht in enger Beziehung zu der allgemeineren Regel 1.12, deren konsequente Interpretation die Regel 1.8 redundant werden lässt.

Regel 1.9 Wenn mehrere Lemmata passen, soll das kürzeste und einfachste verwendet werden.

Eine Definition für „the shortest and simplest name“ wird nicht angegeben. Des Weiteren fällt an dieser Regel auf, dass es Wechselwirkungen mit den Regeln 1.13 und der Regel 1.8 geben kann. (s. u.)

Ausdrücke für Fertigungsverfahren

Die zweite Gruppe der nicht im Lexikon aufgeführten Wörter bilden Bezeichner für Fertigungsverfahren.

Regel 1.10 Verben, die Fertigungsverfahren spezifizieren, dürfen verwendet werden, auch wenn sie nicht im Lexikon enthalten sind.

Regel 1.10 besagt erstens, dass Fertigungsverfahren (Manufacturing Processes) bezeichnende Fachterminologie verwendet werden darf und dass zweitens diese Bezeichner in Verbalkonstrukten verfasst werden müssen.

Regel 1.11 Handelt es sich um polyseme Wörter, deren Wortformen auch anderen Wortarten zugeordnet werden können, dürfen sie nur als Verb verwendet werden, außer wenn das Nomen auch als technischer Bezeichner verwendet werden kann.

Diese Regel stellt eine redundante Pointierung der festgelegten Richtlinien zur Verwendung von Homographen dar. Eine Konsequenz für die maschinelle Verarbeitung von Texten, die gemäß den AECMA SE-Richtlinien erstellt wurden, kann jedoch anhand dieser Regel verdeutlicht werden. Sollte in einem Text ein Wort vorkommen, das nicht in der Wortliste aufgeführt ist, kann durch die Erkennung der Wortart mittels einer syntaktischen Komponente eine grobe semantische Kategorisierung erfolgen. Selbstverständlich kann auch im umgekehrten Fall von einer semantischen Kategorie auf die Wortart geschlossen werden, wobei jedoch die Inferenz der Semantik eines unbekannten Wortes wesentlich aufwändiger ist als das automatische Erkennen der Wortart.

Einheitliche Terminologie

Regel 1.12 Einmal eingeführte Bezeichnungen sollten weiterverwendet werden. Dies gilt insbesondere für Technische Bezeichner.

Während es in anderen Bereichen der schriftlichen Kommunikation erstrebenswert ist, einen möglichst umfangreichen Wortschatz zu verwenden und dabei Wiederholungen zu vermeiden, ist es im AECMA SE erforderlich, einmal verwendete Begriffe konsistent weiterzuverwenden. Die Regel 1.12 stellt eine Verallgemeinerung der Regel 1.8 dar, die aussagt, dass dieselben Entitäten nicht durch unterschiedliche – meist neu eingeführte – Technische Bezeichner benannt werden sollen. Regel 1.12 erweitert diese Anforderung auf im Lexikon enthaltene Wortformen und auf die Fertigungsverfahren beschreibenden Verben.

Regel 1.13 Handlungsanweisungen sind so spezifisch wie möglich zu verfassen.

Diese Regel besitzt eine Wechselwirkung mit der Regel 1.12, da z. B. die Verwendung von Hyperonymen zu einer konsistenteren Terminologie, jedoch zu Lasten der präzisen Formulierung führen kann. Es wird im AECMA SE nicht ausgesagt, welche Regel vorrangig zu beachten ist. Die Zuordnung der Regel zur Gruppe der das Lexikon betreffenden Regeln fokussiert jedoch nur den lexikalischen Aspekt der Spezifizität. Die passende Lexemwahl ist jedoch keine Garantie, da unpräzise Instruktionen auch aus der Syntax resultieren können.

Weitere das Lexikon betreffende Regeln sind lediglich als Hinweise an die Autoren zu verstanden.

Regel 9.2 Wenn Wörter zu Phrasen kombiniert werden, ist zu beachten, dass die im Lexikon definierte Semantik erhalten bleibt.

Regel 9.3 Das Lexikon ist korrekt zu verwenden, um richtige Wörter, Bedeutungen und Wortarten zu erhalten.

Die dem Abschnitt „writing practices“ zugeordneten Regeln 9.2 und 9.3 betonen, dass die im Lexikon definierten Wörter auch wirklich in der dort vorgeschriebenen Bedeutung verwendet werden sollen.

7.2.2.2 Allgemeine Regeln zur Syntax

Während alle die Wortauswahl betreffenden Regeln auf der linguistischen Beschreibungsebene Lexikon anzusiedeln sind, betreffen die folgenden Regel verschiedene Ebenen. Aus diesem Grund werden ab hier explizit die betroffene Beschreibungsebenen – die Ebenen auf denen die Sprachkontrolle erfolgt – zu den Regeln hinzugefügt

Nominalkomposita

Regel 2.1 Die aus dem Prozess der Komposition hervorgegangenen Nomen (noun cluster) dürfen nicht aus mehr als drei Nomen bestehen.

Betroffene Beschreibungsebene: die Annotation der einzelnen Nomen.

Insbesondere innerhalb der technischen Dokumentation ist es üblich Bezeichnungen technischer Namen durch Komposition zu bilden. Neue Komponenten sind nur selten wirklich neu, sondern stellen Verbesserungen existierender Dinge dar. Beides soll sich im Namen wiederspiegeln, so dass häufig ein Kompositum, bestehend aus dem allgemeineren, bisherigen Namen und weiteren Nomen, die die Besonderheit der neuen Komponente kennzeichnen sollen, gebildet werden. Dieser Prozess ist nicht begrenzt und wird in der linguistischen Beschreibung oft durch rekursive Regeln definiert, die die Produktion unbegrenzt langer Komposita erlauben. Die Verarbeitung der Komposita ist jedoch sowohl für Menschen als auch für Maschinen problematisch. Aufgrund des häufigen Auftretens dieser Konstruktionen in den relevanten Textsorten ist ihre Verwendung im AECMA SE trotz der problematischen Verarbeitung zugelassen, jedoch dürfen sie nur aus maximal drei Nomen zusammengesetzt sein. Darüber hinaus unterliegt ihre Verwendung weiteren Restriktionen.

Technische Namen

Regel 2.2 Die Bedeutung von technischen Namen muss verdeutlicht werden.

Betroffene Beschreibungsebene: Wortbedeutung bzw. kompositionale Wortsemantik

Hierfür werden zwei Möglichkeiten angeboten. Zum einen kann mittels Bindestrichen, welche Kompositumsbestandteile gruppieren, eine Disambiguierung erfolgen, zum anderen kann die Bedeutung dieser Bezeichner durch eine bei der Einführung des Begriffs vorkommende Erklärung verdeutlicht werden.

Determinatoren

Regel 2.3 Die Artikel (the, a, an) bzw. die Demonstrative (this, these) sollen verwendet werden.

Betroffene Beschreibungsebene: Syntax des Nominalkomplexes

Eine Angabe über die Zulässigkeit von artikellosen Nominalphrasen wird dergestalt formuliert, dass analog zur Standardsprache, Artikel nicht elliptifiziert werden dürfen.

Gerundium

Regel 3.1 Die Verwendung des Gerundiums (der „-ing“-Form) ist nicht zugelassen.

Betroffene Beschreibungsebene: Lexikon

Die „-ing“-Formen dürfen nicht produktiv verwendet werden, d. h. die zur Verfügung stehenden „-ing“-Formen sind im Lexikon enthalten. Die wenigen im Lexikon enthaltenen Wörter im Gerundium sind lighting, opening, routing und servicing wenn sie als Nomen verwendet werden und mating, missing und remaining als Adjektive. Das einzige im SE-Lexikon enthaltene Wort, welches als Basis für eine produktive Wortbildung des Gerundiums verwendet werden kann, ist das Verb open. Als Konsequenz hieraus ergibt sich u. a., dass im SE die Bildung der Verlaufsformen nicht vorgesehen ist.

Tempus

Regel 3.2 Es dürfen nur die im Lexikon enthaltenen Verbformen verwendet werden.

Betroffene Beschreibungsebenen: Lexikon, Annotation des Tempus

Die zulässigen Verbformen umfassen die Zeitformen: simple present tense, simple past tense und future tense. Die heißt, dass andere Formen (present perfect , past perfect und future perfect) im AECMA Simplified English nicht zugelassen sind.

Des Weiteren sind die Verbformen Imperativ und Infinitiv zulässig.

Verwendung der Partizipien

Regel 3.3 Das past participle darf nur als Adjektiv verwendet werden: sowohl in Kombination mit einem Substantiv oder nach den Verben TO BE, TO BECOME.

Betroffene Beschreibungsebenen: Lexikon, insbesondere die Wortarteninformation, Syntax

Von den vielfältigen Verwendungsweisen der im Lexikon enthaltenen past participle-Formen ist nur ihr attributiver Gebrauch als Adjektiv, sei es vor Substantiven, sei es nach to be oder to become, zugelassen. Diese Regel wird durch weitere Regeln betont.

Regel 3.4 Das past participle darf nicht mit HAVE kombiniert werden um eine nicht legitimierte Zeitform zu bilden.

Betroffene Beschreibungsebenen: Lexikon, Annotation des Tempus

Diese Regel ist in Kombination mit der Tense-Regel (Regel 3.2) und der past participle-Regel (Regel 3.3) zu sehen und durch deren korrekte Anwendung redundant.

Regel 3.5 Das past participle eines Verbs darf nicht mit Hilfe eines Auxilliars (helping verb) in ein komplexes Verb überführt werden.

Betroffene Beschreibungsebenen: Lexikon, Struktur des Verbalkomplexes

Die Verwendung der Begriffe ‚helping verb' und ‚complex verb' geschieht in einer informellen Weise. So werden als Beispiele für ein helping verb "can", "must" und "will" angegeben und als Beispiel für ein nicht zulässiges komplexes Verb dient u. a. "...must be adjusted", d. h. eine Passivform, ein separat beschriebenes Phänomen.

In den Richtlinien zum Simplified English sind des Weiteren im Abschnitt „Verbs" neben den Vorschriften zum Gebrauch des Genus verbi auch Richtlinien zur Lexemauswahl beim Verfassen von Handlungsanweisungen zu finden. Hier werden diese Regeln unter den texttypspezifischen Regeln behandelt.

7.2.2.3 Texttypspezifische Regeln

In technischen Dokumentationen gibt es verschiedene Äußerungsarten. Diese müssen markiert werden, da die Constraints auf ihnen basieren bzw. von ihnen abhängig sind.

Handlungsanweisungen

Die erste der in den Originalrichtlinien des Texttyps Handlungsanweisung betreffenden Regel hat ein formales Kriterium zum Gegenstand:

Regel 5.1 Handlungsanweisungen sollen so kurz wie möglich formuliert werden. Die maximale Satzlänge beträgt 20 Wörter.

Betroffene Beschreibungsebenen: Äußerungsart, Annotation der Wörter

Weitere Regeln beschäftigen sich mit inhaltlichen bzw. semantischen Aspekten der Handlungsanweisung:

Regel 5.2 Pro Satz darf nur eine Anweisung vorkommen.

Betroffene Beschreibungsebenen: Äußerungsart, Annotation der Sätze, Bedeutungseinheiten bzw. Propositionen

Linguistischer formuliert bedeutet diese Regel, dass pro Satz nur eine Proposition ausgedrückt werden darf. Die Regel 5.2 kann jedoch nicht isoliert, sondern muss in Kombination mit Regel 5.3 betrachtet werden.

Regel 5.3 Mehr als eine Anweisung pro Satz darf nur dann vorkommen, wenn mehrere Handlungen gleichzeitig auszuführen sind.

Betroffene Beschreibungsebenen: Äußerungsart, Annotation der Sätze, Bedeutungseinheiten bzw. Propositionen, Semantik (Gleichzeitigkeit)

Eine Anweisung für die gleichzeitige Ausführung mehrerer Handlungen kann als Anweisung für eine komplexe Handlung aufgefasst werden. Eine derartige Sichtweise erlaubt eine widerspruchsfreie Formalisierung der Regeln 5.2 und 5.3.

Ein weiteres formales Kriterium für das Verfassen von Handlungsanweisungen thematisiert die Interpunktion:

Regel 5.5 Wenn eine Handlungsanweisung mit einem beschreibenden Text eingeleitet wird, muss dieser durch ein Komma abgetrennt sein.

Betroffene Beschreibungsebenen: Äußerungsart (Handlungsanweisungen und beschreibende Texte), Annotation der Sätze, Interpunktion

Diese Regel beinhaltet implizit eine Information über die Zulässigkeit der Kombination von Handlungsanweisungen mit beschreibenden Texten. Die Verwendung der Kommata soll Ambiguitäten vermeiden.

Die weiteren Regeln beschäftigen sich mit den Aspekten der Sprachkontrolle auf der Ebene der Syntax.

Eine der in den AECMA-Regeln innerhalb des Abschnittes „Verben“ aufgeführten Regeln, die Regel zum Gebrauch des Genus verbi, ist texttypspezifisch formuliert und deshalb in der Reorganisation aufgeteilt und entsprechend auf die Text- bzw. Satztypen „Handlungsanweisungen“, „Beschreibende Diskurse“ und „Warnungen und Sicherheitshinweise“ bezogen. Die ursprüngliche Bezeichnung wird beibehalten:

Regel 3.6 Handlungsanweisungen müssen im Aktiv stehen.

Betroffene Beschreibungsebenen: Äußerungsart, Genus verbi

Handlungsanweisungen dürfen nicht durch Passivkonstruktionen ausgedrückt werden. Damit soll erreicht werden, dass das Agens der Handlung spezifiziert werden muss. Eine weitere im Abschnitt „Verben“ aufgeführte Regel richtet sich insbesondere an den Prozess der Formulierung von Handlungsanweisungen.

Regel 3.7 Wenn ein zulässiges Verb für die Beschreibung einer Handlung zur Verfügung steht, soll es benutzt werden.

Betroffene Beschreibungsebene: Verbsemantik

Diese Regel besagt, dass Handlungen, die auch mit verschiedenen Wortarten verbalisiert werden können, durch Verben ausgedrückt werden sollen. Diese Regel interagiert mit den Lexikonregeln.

Die die handlungsanweisungsspezifizierenden Regeln abschließende Vorschrift widmet sich dem Sprechakt dieses Texttyps:

Regel 5.4 In Handlungsanweisungen müssen die Verben in ihrer Imperativform geschrieben werden.

Betroffene Beschreibungsebenen: Äußerungsart, Verbmodus

Die Verwendung des Imperativs der im Lexikon aufgeführten Wörter ist durch die Regel 3.2 legitimiert. Aus dieser Regel ergibt sich, dass das grammatische Subjekt in Handlungsanweisungen nicht vorkommt. Das Agens bleibt jedoch – im Gegensatz zu im Passiv formulierten Anweisungen – in Imperativkonstruktionen spezifiziert, da sich das elliptifizierte Subjekt auf den Leser bzw. den Hörer des Satzes bezieht.

Beschreibende Diskurse

Auch die Richtlinien für beschreibende Texte beginnen mit relativ einfach formalisierbaren Kriterien von Sätzen.

Regel 6.1 Beschreibungen sollen so kurz wie möglich formuliert werden. Die maximale Satzlänge beträgt 25 Wörter.

Betroffene Beschreibungsebenen: Äußerungsart, Annotation der Wörter

Die Regel ist mit der entsprechenden Regel für Handlungsanweisungen vergleichbar, wobei deren maximale Satzlänge um 20% kürzer ist. Dieser Unterschied lässt darauf schließen, dass davon ausgegangen wird, dass Handlungsanweisungen generell kürzer sein sollen.

Regel 6.2 Die Satzlänge soll variiert werden um die Texte interessanter zu gestalten.

Betroffene Beschreibungsebenen: Annotation der Wörter, Annotation der Sätze

Regel 6.3 Die Struktur und die logische Gliederung der Texte soll durch die Verwendung von Absätzen verdeutlicht werden.

Betroffene Beschreibungsebenen: Annotation der Textstruktur, Semantik

Der Aufbau und der Inhalt der Abschnitte (Teil der Semantik) wird weiter reglementiert:

Regel 6.4 Jeder Abschnitt darf nur ein Thema beinhalten.

Betroffene Beschreibungsebenen: Annotation der Textstruktur, Annotation des Themas

Die logische Gliederung der Abschnitte erfolgt insbesondere dadurch, dass nur ein Thema pro Absatz behandelt wird. Auch wenn nicht ausgeschlossen ist, dass ein Thema über mehrere Absätze behandelt wird, kann durch diese Regel das Erkennen von Abschnittsgrenzen als Indiz für einen Themenwechsel oder eine Fokusverschiebung gesehen werden.

Eine XML-basierte Überprüfung der thematischen bzw. rhetorischen Struktur kann durch die Annotation der durch die rhetorical structure theory (RST, Mann und Thompson, 1988) formalisierten rhetorischen Strukturen erfolgen. Es wurde gezeigt, dass diese Theorie eine sehr gute Basis ist, um Textzusammenhänge mit Auszeichnungssprachen zu modellieren. (Lobin 1999c, Rehm 1999)

Regel 6.5 Der Abschnitt muss mit einem das Thema absteckenden Satz eingeleitet werden.

Betroffene Beschreibungsebenen: Annotation der Textstruktur, Annotation des Themas, Rhetorische Struktur

In den Exemplifizierungen zum AECMA-SE ist zu sehen, dass dieser einleitende Satz quasi den Status einer Kurzzusammenfassung besitzt. Hierdurch wird es einerseits ermöglicht das behandelte Thema schnell zu erfassen, andererseits können bereits die wichtigsten Begriffe und Schlüsselwörter eingeführt werden.

Regel 6.6 Die Verbindung der Sätze mit dem Thema soll durch Schlüsselwörter verdeutlicht werden.

Betroffene Beschreibungsebenen: Annotation des Themas, Wortbedeutung (der Schlüsselwörter)

Diese Schlüsselwörter sind in der den Abschnitt einleitenden Zusammenfassung eingeführt worden. Auf sie sollte im Verlauf des logisch zusammenhängenden, beschreibenden Textes durch deiktische Ausdrücke – wie z. B. Demonstrativpronomina – verwiesen werden.

Regel 6.7 Ein Abschnitt darf aus höchstens 6 Sätzen bestehen. Innerhalb einer Sequenz von 10 Absätzen darf maximal ein Abschnitt aus nur einem Satz bestehen.

Betroffene Beschreibungsebenen: Annotation der Textstruktur, Annotation der Wörter

Neben den oben aufgeführten inhaltlichen Regeln zur Verwendung von Absätzen wird in dieser Regel das formale Kriterium der Abschnittsgröße festgelegt. Diese Regel besitzt denselben Status wie die Regeln zur Satzlänge und sollte auf eine ähnliche Art formalisiert werden.

Regel 6.8 Neue und komplexe Informationen sollen langsam eingeführt werden.

Betroffene Beschreibungsebene: Semantik

Diese Regel zählt zu den sehr schwer formalisierbaren Regeln, da kein genereller Bewertungsmaßstab für „neue und komplexe Informationen“ existiert.

Nur eine Regel beschäftigt sich mit den speziellen syntaktischen Aspekten der Handlungsanweisungen, die oben bereits angesprochene Regel zur Verwendung des Genus verbi. Diese Regel wird ebenfalls mit der ursprünglichen

Bezeichnung „Regel 3.6“ wiedergegeben, obwohl sie natürlich wiederum nur einen texttypbezogenen Ausschnitt enthält:

Regel 3.6 Beschreibende Texte sollen in einem möglichst großen Umfang im Aktiv formuliert werden.

Betroffene Beschreibungsebene: Genus verbi

Diese Regel besagt indirekt, dass es zulässig ist, beschreibende Texte auch im Passiv zu formulieren, auch wenn hierauf in den Ausführungen zum AECMA-Regelwerk nicht weiter eingegangen wird. In den Richtlinien wird der Zielgruppe, den technischen Redakteuren, gezeigt, wie Passiv-Aktiv-Transformationen durchgeführt werden können. In dieser Beschreibung wird jedoch nur der Fall thematisiert, in dem das Subjekt eines im Aktiv formulierten Satzes im Passiv Bestandteil einer Präpositionalphrase ist. Dies ist jedoch nicht der Normalfall, da Passivkonstruktionen sehr häufig deshalb verwendet werden um diesen – im Aktiv obligatorischen Bestandteil des Satzes – weglassen zu können. Dies ist im AECMA-SE möglich, wenn auch nicht empfohlen.

Warnungen und Sicherheitshinweise

Die letzte Gruppe der in den Richtlinien angesprochenen Diskurstypen beinhaltet Textteile, die besonders hervorzuheben und verständlich formuliert sein müssen: die Aufforderungen zur Vorsicht. Insbesondere semantische und textsatzbezogene Punkte werden hierfür reglementiert.

Die allgemeinste und am schwierigsten formalisierbare Regel lautet:

Regel 7.2 Warnungen und Sicherheitshinweise müssen präzise formuliert sein.

Betroffene Beschreibungsebenen: Äußerungsart , Semantik

Eine spezifischere Regel beinhaltet Hinweise für den Beginn einer Äußerung:

Regel 7.1 Warnungen und Sicherheitshinweise müssen mit einer klaren Anweisung beginnen.

Betroffene Beschreibungsebenen: Äußerungsarten, Verbmodus (Imperativ)

Zwei weitere Regeln sprechen die Thematiken an, die u. U. in Warnungen angesprochen werden müssen.

Regel 7.5 Wenn Bedingungen erfüllt sein müssen, bevor bestimmte Handlungen ausgeführt werden können, müssen diese der Warnung bzw. dem Sicherheitshinweis vorangestellt werden.

Betroffene Beschreibungsebenen: Äußerungsart, Kausalzusammenhänge

Regel 7.3 Wenn nötig, sind den Warnungen und Sicherheitshinweisen klare Angaben über mögliche Risiken hinzuzufügen.

Betroffene Beschreibungsebenen: Äußerungsart , Semantik

Eine die Formatierung gedruckter bzw. elektronisch präsentierter Texte betreffende Regel weist darauf hin, dass Warnungen und Sicherheitshinweise auch optisch hervorgehoben sein sollen.

Regel 7.4 Warnungen und Sicherheitshinweise müssen eindeutig gekennzeichnet sein.

Betroffene Beschreibungsebenen: Äußerungsarten

Für das Anfertigen von Sicherheitshinweisen und Warnungen sind keine eigenen syntaktischen Reglementierungen in den Richtlinien enthalten. Jedoch gibt es die bereits aufgeteilt dargestellte Regel zum Genus Verbi, deren Gültigkeit weiterhin besteht, sowie die Regel zu den imperativisch zu formulierenden Handlungsanweisungen. Bei einer genaueren Betrachtung des Wesens von Sicherheitshinweisen und Warnungen fällt auf, dass – entgegen der häufig vorgenommenen strikten Einteilung eines Satzes in die Gruppe Anweisung, Beschreibung oder Warnung (vgl. Lehrndorfer 1996:155) – die Warnung ihrerseits beschreibende oder anweisende Phrasen enthalten kann. Dies lässt sich an einer in den AECMA-Richtlinien enthaltenen Beispielwarnung verdeutlichen:

> WARNING: Make sure that the oxygen tubes are fully clean. Oxygen and oil or grease make an explosive mixture. An explosion can cause death or injury to personnel and / or damage to equipment.

Diese Warnung besteht aus einer einleitenden Imperativkonstruktion und zwei beschreibenden Sätzen. Das bedeutet, dass alle weiteren Regeln zu den entsprechenden Diskurstypen erhalten bleiben. Für die Formalisierung hat dies zur Folge, dass nicht von einer direkten Beziehung von Textteilen zu Diskurstypen ausgegangen werden kann, sondern sich auch komplexere Zuordnungen ergeben können.

Die in der Regel 7.4 vorgeschriebene Kennzeichnung der Äußerungsart gilt jedoch insbesondere der Publikation des Textes. Eine korrekte Annotation der betroffenen Textteile erlaubt es aber, diesen Aspekt in den Hintergrund treten zu lassen, da die zu veröffentlichenden Versionen der Texte durch die Verwendung von Formatierungssprachen erzeugt werden.

7.2.2.4 Allgemeine – sprechaktübergreifende Regeln

Wie sich schon bei den sprechaktbezogenen Regeln des AECMA-SE zeigte, besteht ein Mittel der Sprachkontrolle darin, die Möglichkeiten der Informationsverteilung auf formale sprachliche Einheiten zu beschränken. So besagt Regel 5.2, dass pro Satz nur eine Anweisung vorkommen darf, und eine Vielzahl der Regeln für beschreibende Texte beschäftigen sich mit der Verteilung von Informationseinheiten in beschreibenden Texten auf Sätze und Abschnitte. Eine die Inhaltsorganisation für alle Sprechakte generalisierende Regel lautet:

Regel 4.1 Pro Satz darf nur ein Thema (bzw. topic) behandelt werden.

Betroffene Beschreibungsebenen: Thema bzw. Topic, Annotation der Sätze

Diese Regel erfordert eine relativ abstrakte, semantische Kontrolle. Eine auf den ersten Blick möglicherweise naheliegende Gleichsetzung der Ebene Thema und der Ebene Proposition kann nicht erfolgen, da eine Thematik durchaus mittels mehrerer Propositionen versprachlicht werden kann. Dies zeigt sich z. B. innerhalb des Regelwerks zu den Anweisungen. Im Falle der Notwendigkeit einer zeitgleichen Ausführung mehrerer Arbeitsschritte werden die unterschiedlichen Anweisungen durch einen Konjunktor verbunden in einem Satz präsentiert. Ein Thema - mehrere Propositionen.

Eine weitere Regel hilft einen etwas holprigen Stil zu vermeiden, der sich ergeben würde, wenn eine Reihe von Ein-Thema Sätzen aneinandergereiht wird:

Regel 4.4 Der Bezug von aufeinander folgenden, logisch zusammenhängenden Sätze soll mit Konnektoren verdeutlicht werden.

Betroffene Beschreibungsebenen: Annotation der Sätze, Kohärenz, Wortbedeutung (Konnektoren)

Aus dieser Regel folgt, dass – abweichend von standardsprachlichen Druckschriften – in im AECMA-SE verfassten Texten sehr viele Sätze mit Konnektoren beginnen.

Die im Bereich der technischen Dokumentation üblichen Praxis der Verkürzung von Sätzen durch Auslassungen ist im AECMA-SE nicht vorgesehen:

Regel 4.2 Es dürfen keine Wörter weggelassen werden um Sätze kürzer werden zu lassen.

Betroffene Beschreibungsebene: Syntax

In den Beispielen zu dieser Regel wird das Auslassen von Nomen und Verben angesprochen. So wie sie formuliert wurde, bezieht sie sich jedoch auch auf andere Wortarten. Zusätzlich wird in einer allgemeinen Empfehlung am Ende des Regelwerks eine nichtnummerierte Regel eingefügt, die separat betont, dass die Konjunktion „that“ im AECMA-SE nicht ausgelassen werden darf.

7.2.3 Kontrolliertes Deutsch

In diesem Abschnitt werden Regeln vorgestellt, die aus zwei Publikationen stammen und die skizzieren wie die Erstellung deutschsprachige Texte kontrolliert werden können. Eine der Quellen (Schachtl, 1996) beschreibt das sogenannte Siemens Dokumentationsdeutsch (SDD), die andere (Lehrndorfer, 1995) skizziert ein Kontrolliertes Deutsch, indem sie ‚linguistische und sprachpsychologische Leitlinien für eine (maschinell) kontrollierte Sprache in der technischen Dokumentation’ – so der Untertitel ihrer Monographie – aufstellt.

Das Siemens Dokumentationsdeutsch klassifiziert die Regeln durch zwei Arten von Anweisungen. Die erste Gruppe – die editierspezifischen Regeln – stellt Regeln zusammen, die sich entweder ausschließlich mit den Aspekten der Dokumentstruktur befassen oder mit deren Interaktion mit dem textuellen Inhalt, während der zweite Typ Regeln – die Syntaxregeln – nur die sprachlichen, insbesondere syntaktischen Regeln enthält. Auch diese Regeln sollen hier wiedergegeben und um Verweise auf die betroffenen linguistischen und außerlinguistischen Ebenen erweitert werden.

Editierspezifische Regeln:

1. Im Deutschen unterscheidet sich die Syntax von Überschriften von der Syntax von Absatztexten. Deshalb müssen Überschriften markiert werden.
2. Die Grammatik des SDD benötigt eine klare Auszeichnung von Objektcode. Wenn z. B. auf die Beschriftungen von Geräten, Tasten o. ä. Be-

zug genommen wird, muss dies in der Bedienungsanleitung deutlich erkennbar sein.

3. Bei einer Liste, z. B. einer Aufzählung können entweder die einzelnen Aufzählungselemente mit dem Fließtext zusammen interpretiert werden oder jedes einzelne Listenelement enthält einen vollständigen, syntaktisch korrekten Text, d. h. einen ganzen Satz oder mehrere Sätze.
4. Die Verwendung von Klammern ist dergestalt restringiert, dass nur syntaktisch korrekte Phrasen oder Wörter in Klammern stehen dürfen. Durch diese Regel wird z. B. die Anrede ‚Liebe(r) Leser(in)' untersagt.
5. Die letzte der Regeln thematisiert die Verwendung von Abkürzungen. Das Interpunktionszeichen wird einerseits als Abkürzungs- andererseits aber auch als Satzendemarkierung verwendet, bzw. – wenn die Abkürzung am Satzende steht – für beide Zwecke. Die Lösung zur Ambiguitätsauflösung besteht für SDD darin, dass bestimmte Abkürzungen nie am Ende eines Satzes (Beispiele: z.B., d.h., s.) oder ausschließlich am Ende (etc., usw.) zu verwenden sind.

Die Kontrolle der Einhaltung der editierspezifischen Regeln kann z. T. auf der Auszeichnung der Textstruktur basieren. Die Annotation der Textstruktur ist in der Praxis der Erstellung von Technischen Dokumentationen in sehr vielen Fällen bereits zu finden, da diese als Grundlage der Erstellung der Papierversion der Texte dient. Die Regel 1 erfordert die Auszeichnung von Überschriften (in HTML z. B. als `<H2>`), Regel 2 die Annotation des Objektcodes (in einigen HTML-Versionen z. B. als `<code>`) und Regel 3 die Markierung von Listen. Die Ebene der Textstruktur interagiert hier mit der Ebene der Syntax, z. B. mit einer Auszeichnung von kompletten oder elliptischen Sätzen. Für die Überprüfung der Regeln 4 und 5 erscheint es sinnvoll, das klassische Annotationsinventar um weitere zumindest konzeptuelle Annotationsebenen zu erweitern. Zum Einen sollten die Klammern ausgezeichnet werden (Regel 4), zum Anderen die Bedeutung der Punkte (Regel 5). Darüber hinaus kann auch erwogen werden, die Abkürzungen selbst zu annotieren (Regel 5). Es soll an dieser Stelle erwähnt werden, dass durch die Verwendung einer Annotation der Satzende- und der Abkürzungspunkte, gegebenenfalls ergänzt um die Auszeichnung der Abkürzungen selbst, natürlich, wie in den vorhergenannten Fällen einerseits, die Einhaltung des kontrolliertsprachlichen Regelwerks mit den Mitteln von Dokumentgrammatiken überprüft werden kann, dass jedoch andererseits die Regel 5 selbst irrelevant wird. Der Zweck dieser Regel be-

steht darin, einem Sprachverarbeitungssystem die Aufgabe der Disambiguierung zu erleichtern – ein Prozess, der durch die Annotation irrelevant wird.

Syntaxregeln

1. Da das Deutsche eine relativ freie Wortstellung besitzt und unterschiedliche Kasusformen häufig nicht an den Nomen bzw. Artikeln erkennbar sind, treten in Sätzen oft Subjekt-Objekt-Ambiguitäten auf. Um diese maschinell disambiguieren zu können muss das Subjekt immer vor dem Objekt bzw. den Objekten stehen.[94]
2. Es dürfen nur max. zwei Präpositionalphrasen direkt hinter einander auftreten.
3. Konditionale Sätze müssen 'wenn' verwenden.
4. Koordiniert werden dürfen: Nominalphrasen, Adjektivphrasen und Präpositionalphrasen sowie Sätze.
5. Ellipsen dürfen in koordinierten Konstruktionen verwendet werden wobei die folgenden Restriktionen gelten:
    a. sie dürfen nicht im ersten Konjunkt auftreten
    b. es darf nur das Subjekt oder das finite Auxilliar elliptifiziert werden.

Diese Regeln kontrollieren die Syntax auf sehr unterschiedlichen Ebenen. Analog zur Erwähnung der „Ebenen der Kontrolle“ zu den im Abschnitt 7.2.2 dargestellten Regeln zur Syntax des AECMA SE seien hier diese Ebenen für die Syntaxkontrolle des SDD angegeben.

Regel 1 erfordert die Annotation der funktionalen Struktur, zumindest die Annotation des Subjekts und der Objekte. Darüber hinaus muss die Phrasenstruktur, annotiert werden, d. h. zumindest müssen die vorkommenden Präpositionalphrasen (für die Regeln 2 und 4) und die Nominal- und Adjektivphrasen ausgezeichnet werden (Regel 4). Die Überprüfung der dritten Regel mit den Mitteln von Auszeichnungssprachen erfordert eine detaillierte Annotation der Nebensätze. Für die Kontrolle der Regeln 3, 4 und 5 müssen auch die Wortarten (zumindest die Konjunktionen) bekannt sein.

---

[94] Die Wissen über die Einhaltung dieser Regel erlaubt einer maschinellen Verarbeitung den Ausschluss von Interpretationen. Den Menschen, die einen derartig restringierten Text lesen, erleichtert die Kontrolle der Syntax mit dieser Regel das Textverständnis jedoch nicht, da die Rezipienten der Texte normalerweise keine Kenntnis der syntaxkontrollierenden Regelwerke besitzen.

Neben diesen wenigen aber relativ präzisen Regeln des SDD sind weitere Regeln, die für eine Kontrolle des Deutschen relevant sein können, in den Leitlinien für ein kontrolliertes Deutsch (Lehrndorfer 1995) publiziert worden. Eine der wichtigsten Grundlagen für ihr Regelwerk bildet das in Abschnitt 7.2.2 ausführlich dargestellte AECMA Simplified English. Einzelne der Regeln werden direkt übernommen, andere jedoch verworfen. Nachfolgend sollen kurz zwei Regeln aufgeführt werden, die als eigenständig aufgefasst werden können. Diese Eigenständigkeit ist zweierlei Ursprungs: die erste aufgeführte Regel Lehrndorfers für ein Kontrollierte Deutsch widerspricht der entsprechenden AECMA-Regel für ein Kontrolliertes Englisch, wohingegen der zweite reglementierte Aspekt der Sprache im AECMA-Regelwerk nicht thematisiert wurde, da er für das Englische keine Relevanz besitzt.

Lehrndorfer erlaubt die Elliptifizierung von Artikeln in Nominalphrasen. Dies widerspricht in gewisser Weise einem der Kriterien der Kontrollierten Sprache überhaupt, da die Kontrolle nur syntaktisch korrekte Sätze betreffen soll. Im AECMA-Regelwerk wird dies auch durch die Regel 4.2 explizit gefordert. Das Siemens Dokumentationsdeutsch erlaubt derartige Konstruktionen, jedoch dürfen sie nur in Überschriften verwendet werden. Lehrndorfer lässt sie an allen Stellen zu, da sie das Textverständnis erleichtern.

Die Zulassung elliptischer und nichtelliptischer Nominalphrasen besitzt allerdings den Nachteil, dass für beide Konstruktionen Regeln spezifiziert werden müssen. Kann bei der Überprüfung der Regelwerke jedoch auf die Annotation syntaktischer Informationen zurückgegriffen werden und werden die elliptifizierten Wörter – hier die Artikel – in einer Annotationsebene z. B. durch die Verwendung leerer Elemente markiert, dann ist es möglich, dass für die maschinelle Überprüfung der Syntax ein (leicht angepasstes) Regelwerk ausreicht.

Die angesprochene sprachspezifische Regel besagt, dass im Kontrollierten Deutsch Verbklammern vermieden werden sollen. Die Überprüfung dieser Regel kann durch die Annotation der Satzfelder ermöglicht werden. Diese Annotation ist im Abschnitt 6.2.6 thematisiert worden. Ist eine solche Auszeichnung vorhanden, kann ein komplexer Schemasprachenausdruck[95] dafür sorgen, dass entweder nur der als linke Verbklammer oder nur der als rechte Verbklammer bezeichnete Teil des Verbalkomplexes vorhanden sein muss. Eine derartige Überprüfung kann darüber hinaus auch zur Überprüfung der

[95] Inwieweit dies möglich ist, ist natürlich auch von der Mächtigkeit der gewählten Schemasprache abhängig. (vgl. Abschnitt 2.4)

Korrektheit der Syntax verwendet werden, wenn auf die Annotation der Satz- bzw. Teilsatztypen bezug genommen wird: in Nebensätzen muss die linke Verbklammer fehlen, wohingegen in Hauptsätzen eher die rechte Verbklammer fehlen kann.

Lehrndorfer diskutiert in ihren Leitlinien für ein Kontrolliertes Deutsch neben den syntaktischen Regeln auch das Lexikon. Nach einer Abwägung der positiven und negativen Aspekte der Bereitstellung eines Lexikons entschied sie sich dafür, ein Lexikon für ein Kontrolliertes Deutsch zusammenzustellen. Im Gegensatz zum AECMA-Lexikon handelt es sich um ein Stammformlexikon. Dieses Lexikon kann jedoch mit texttechnologischen Mitteln in ein Vollformlexikon überführt werden. Eine derartige Komponente wurde bereits implementiert und beschrieben. (vgl. Pönninghaus, 2000; Pönninghaus und Witt 2001)

Ausblick

Der Ausblick soll mit einem Rückblick beginnen:

> Den Ausgangspunkt der vorliegenden Untersuchung bildete der Wunsch, im Zusammenhang mit der Entstehung der Auszeichnungssprachen entwickelte Techniken für sprachtechnologische Anwendungen zu verwenden. Hierbei war in erster Linie daran gedacht, standardisierte linguistische Annotationen von Texten mit Auszeichnungssprachen als gegeben anzunehmen. Konkret fokussiert werden sollte hierbei die Fragestellung, inwieweit die den Auszeichnungssprachen inhärenten Möglichkeiten der Strukturüberprüfung, die normalerweise verwendet werden um den Aufbau von zu publizierenden Texten zu kontrollieren, ausgenutzt werden können, um sprachliche Ausdrucksmöglichkeiten zu beschränken.
>
> Vorliegender Text, Seite 6

Paraphrasiert und konkretisiert besagen diese Zeilen, dass am Anfang dieser Arbeit der Wunsch stand, eine maschinelle Überprüfung der Einhaltung kontrolliertsprachlicher Regelwerke mit der Hilfe von Dokumentgrammatiken (z. B. DTDs) zu erlauben.

In der Vorstellung und Diskussion der im Abschnitt 7.2 aufgeführten Regeln von Kontrollierten Sprachen ist herausgearbeitet worden, dass diese Regeln Restriktionen der Standardschriftsprache vornehmen, die äußerst unterschiedliche linguistische und nichtlinguistische Ebenen betreffen. Im einzelnen wurden u. a. die folgenden linguistischen Ebene herausgearbeitet: Lexikon, Annotationen von (ausgewählten) Wortarten, Wortsemantik, Syntax von Nominalphrasen und Tempus. Hinzu kamen quantitative Kriterien wie die Satz- oder Absatzlänge und die Ebene der Textpublikationskriterien z. B. die Annotation von Überschriften, Aufzählungen oder auch der Interpunktionszeichen.

An dieser Stelle hätte aufgesetzt werden können, wenn Dokumentgrammatiken zum Zwecke der Einhaltung dieser Regeln erarbeitet worden wären – dem ursprünglichen Ziel dieser Arbeit. Allerdings war die Herausarbeitung dieser Ebenen nicht die einzige Voraussetzung, um dieses Ziel zu erreichen, vielmehr – und viel wichtiger – musste es erst gezeigt werden, wie derartig unterschiedliche Ebenen in XML-basierten Formalismen repräsentiert werden können. Dies ist durch die vorliegende Untersuchung, Analyse und Weiterentwicklung derartiger Auszeichnungssprachen erreicht worden. Mit der vor-

gelegten Arbeit steht jetzt hierfür eine Basisarchitektur zur Verfügung, d. h. auf dieser theoretischen Fundierung kann aufgesetzt werden.

Als mögliche Anwendungen sind jedoch keineswegs nur Kontrollierte Sprachen erkennbar, vielmehr ist das hier entwickelte Repräsentationsmodell für die verschiedensten, d. h. insbesondere auch nichtlinguistischen Anwendungen nutzbar. Derartige unterschiedlichen Anwendungen sollen die Gegenstände weiterer Arbeiten sein.

Neben der Bereitstellung eines Fundaments für die anwendungsorientiertere Verwendung von Auszeichnungssprachen bietet das entwickelte Modell aber auch auf der theoretischen Seite eine Vielzahl von Anknüpfungspunkten, wobei ein Punkt besonders hervorgehoben werden soll: Durch die multiple Annotation auf der primären Ebene der Dokumentstrukturierung ergeben sich zwangsläufig auch auf der sekundären Ebene völlig neue Untersuchungsperspektiven, Perspektiven die bereits Gegenstand weiterer Arbeiten sind.

Literatur

Vorbemerkung

Eine Vielzahl der nachfolgend aufgeführten Literaturangaben wurde parallel oder z. T. sogar ausschließlich im Internet veröffentlicht. Die über das Netz verfügbare Literatur wurde allerdings nicht besonders gekennzeichnet, insbesondere wurde auch auf die Angabe der Internetadresse (des URI) verzichtet. Für dieses Vorgehen sind mehrere Gründe zu nennen:

Erstens besteht das seit langem bekannte und häufig thematisierte Problem, dass viele dieser Adressangaben nur eine begrenzte Zeit Gültigkeit besitzen. Verschiedene neuerer Zitierkonventionen (z. B. die des Duden-Verlags) schlagen vor diesem Problem damit zu begegnen, dass angegeben werden soll, wann die Gültigkeit der Adressen letztmalig überprüft wurde. Der primäre Zweck eines jeden Literaturverzeichnisses besteht allerdings darin, den Rezipienten des zitierenden Textes die Möglichkeit zu geben, die Quellen konsultieren zu können. Da es jedoch keine mit den Bibliotheken vergleichbare Organisation gibt, die mit der Hilfe einer Internetadresse und einer Datumsangabe die zitierte Primärliteratur rekonstruieren kann, wird diesem Zweck durch eine derartige Zitierkonvention nicht nähergekommen.

Der zweite Grund besteht darin, dass bereits die gegenwärtig vorhandenen Suchmaschinen so erfolgreich arbeiten, dass die Angaben der im Literaturverzeichnis aufgeführten Informationen zum Finden der Quellen völlig ausreichen. Hinzu kommt, dass einige der Suchmaschinen bis zu einem gewissen Punkt eine oben angesprochene klassische Bibliotheksaufgabe übernommen haben: Sie verweisen nicht nur auf die Quellen sondern sie fertigen auch eine Kopie an, die konsultiert werden kann, wenn die originale Ressource nicht mehr verfügbar ist.

Als weiterer Grund soll angeführt werden, dass immer mehr Literatur, insbesondere Zeitschriftenaufsätze, digitalisiert vorliegen oder in eine elektronische Version überführt werden, die jedoch nur einer kleinen Gruppe von Benutzern des Netzes zugänglich gemacht werden.

Abiteboul, Serge, Peter Buneman und Dan Suciu (2000). Data on the Web : from relations to semistructured data and XML. San Francisco: Kaufmann.

Altheim, Murray und Shane McCarron (Hg., 2001). XHTML 1.1 - Module-based XHTML. Mai 2001, W3C

Barnard, David T., Lou Burnard, Jean-Pierre Gaspart, Lynne A. Price, C.M. Sperberg-McQueen, Giovanni Battista Varile (1995). Hierarchical Encoding of Text: Technical Problems and SGML Solutions. In: Ide und Véronis (Hg., 1995), 211-231

Barthe, Kathy, Claire Juaneda, Dominique Leseigneur, Jean-Claude Loquet, Claude Morin, Jean Escande und Annick Vayrette (1999). GIFAS Rationalized French: A Controlled Language for Aerospace Documentation in French. In: Technical Communication 2/1999. S. 220-229.

Baumann, Klaus-Dieter: Textuelle Eigenschaften von Fachsprachen. In: Hoffmann et al. (1998). S. 408-416

Bausani, Alessandro (1970). Geheim- und Universalsprachen: Entwicklung und Typologie, Stuttgart: Kohlhammer.

Bird, Steven und Mark Liberman (2001). A formal framework for linguistic annotation. In: Speech Communication Vol. 33 (1-2), S. 33-60

Bray, Tim, Dave Hollander und Andrew Layman (Hg., 1999). Namespaces in XML. Januar 1999, W3C.

Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen und Eve Maler (Hg., 2000). Extensible Markup Language 1.0 (Second Edition). Oktober 2000, W3C.

Bresnan, Joan, (Hg., 1982). The Mental Representation of Grammatical Relations. Cambridge, Massachusetts: MIT Press.

Bresnan, Joan. (2001). Lexical-Functional Syntax . Oxford: Blackwell.

Brüggemann-Klein, Anne (1993). Formal Models in Document Processing. Habilitationsschrift. Universität zu Freiburg i.Br. Freiburg.

Carpenter, Bob (1992). The Logic of Typed Feature Structures. Albany: Cambridge University Press.

Chomsky, Noam (1957). Syntactic Structures. The Hague: Mouton.

Clark, James (Hg., 1999). XSL Transformations (XSLT) Version 1.0. November 1999. W3C.

Clark, James und Steve DeRose (1999). XML Path Language (XPath). Version 1.0. November 1999, W3C.

Clark, James und Makoto Murata (2001a). "RELAX NG Specification", Committee Specification, August 2001, OASIS.

Clark, James und Makoto Murata (2001b). "RELAX NG DTD Compatibility Annotations", Working Draft, August 2001, OASIS.

DeRose, Steven (1997). The SGML FAQ Book: Understanding the Foundation of HTML and XML. Dordrecht: Kluwer.

DeRose, Steven (1999). XML and the TEI. In: Computers and the Humanities. 33(10). 1999. S. 11-30

DeRose, Steve, Eve Maler und Ron Daniel Jr. (Hg., 2001), XML Pointer Language (XPointer) Version 1.0, W3C.

Durand, David, Elli Mylonas und Steven DeRose. 1996. What Should Markup Really Be? Applying theories of text to the design of markup systems. ALLC/ACH 1996 Bergen, Norwegen.

Dybkjær, Laila (2000). Final Report. MATE Deliverable D6.2. Natural Interactive Systems Lab, University of Southern Denmark.

Dybkjær, Laila, Niels Ole Bernsen, Hans Dybkjær, David McKelvie und Andreas Mengel (1998). The MATE Markup Framework. MATE Deliverable D1.2. Natural Interactive Systems Lab, University of Southern Denmark.

Einstein, Leopold (1885): Zur Geschichte der Weltsprachlichen Versuche von Leibnitz bis auf die Gegenwart. In: Bayrische Lehrerzeitung. München. Nachdruck in Haupenthal (1976)

Friedl, Jeffrey (1998): Mastering regular expressions. Peking: O'Reilly.

Roger Garside, Geoffrey Leech and Tony McEnery (Hg., 1997). Corpus Annotation: Linguistic Information from Computer Text Corpora. London: Longman.

Gazdar, Gerald und Chris Mellish (1989). Natural language processing in PROLOG: an introduction to computational linguistics. Wokingham: Addison-Wesley.

Gippert, Jost (1999), Language-specific encoding in multilingual corpora: Reqirements and solutions. In: Gippert (Hg., 1999), S. 372 - 384.

Gippert, Jost (Hg., 1999), Multilinguale Corpora - Codierung, Strukturierung, Analyse. Prag: enigma corporation.

Goldfarb, Charles F. (1990). The SGML handbook. Oxford: Clarendon Press.

Göpferich, Susanne (1998). Interkulturelles Technical Writing. , Tübingen: Gunter Narr Verlag.

Görz, Günther (Hg., 1993). Einführung in die künstliche Intelligenz. Bonn: Addison-Wesley.

Grosso und Veillard (Hg., 2001). XML Fragment Interchange. W3C Candidate Recommendation. 12 February 2001.

Grishman, R., 1997. TIPSTER Architecture Design Document Version 2.3. Technical Report, DARPA, 1997

Haupenthal, Reinhard (Hg., 1976). Plansprachen: Beiträge zur Interlinguistik. Darmstadt: Wissenschaftliche Buchgesellschaft.

Haugeneder, H. und H. Trost (1993). Beschreibungsformalismen für sprachliches Wissen. In: G. Görz (Hg., 1993). S. 372 - 424.

Hellwig, Peter (1986). Dependency Unification Grammar. In: Proceedings of COLING 86, Bonn. S. 28-33

Heyer, Gerhard und Christian Wolff (Hg., 1998). Linguistik und neue Medien. Wiesbaden: DUV.

Hoffmann, Lothar (1998). Syntaktische und morphologische Eigenschaften von Fachsprachen. In: Hoffmann et al. (1998). S. 416-427.

Hoffmann, Lothar, Hartwig Kalverkämper und Herbert Ernst Wiegand (1998). Fachsprachen / Languages for Special Purposes, Handbücher zur Sprach- und Kommunikationswissenschaft Vol. 14, Berlin: de Gruyter.

Huijsen, Willem-Olaf (1998). Controlled Language – An Introduction. In: Proceedings of the Second International Workshop on Controlled Language Applications (CLAW 98), S. 1-15.

Ide, Nancy und Jean Véronis (Hg., 1995). Text Encoding Initiative: Background and Context. , Dordrecht: Kluwer.

Ide, Nancy (1998). Corpus Encoding Standard: SGML Guidelines for Encoding Linguistic Corpora. Proceedings of the First International Language Resources and Evaluation Conference. Granada, Spanien. S. 463-70.

ISO/IEC 8859 (1986). Standard Generalized Markup Language. Genf: International Organization for Standardization.

ISO/IEC 10646 (2000). The Unicode Standard, Version 3.0. Reading, Massachusetts: Addison-Wesley.

ISO/IEC 10179 (1996). Document Style Semantics and Specification Language. Genf: International Organization for Standardization.

ISO/IEC 10744 (1997). Information processing-Hypermedia/Time-based Structuring Language (HyTime) - 2d edition. Genf: International Organization for Standardization.

Jelliffe, Rick (2001). "Schematron", Academia Sinica. Internetdokument, September 2001.

Jespersen, Otto (1928). Eine internationale Sprache. Heidelberg: Winter.

Johansson, Stig (1995). The Encoding of Spoken Text. In: Ide und Véronis (Hg., 1995), S. 149-158.

Kamp, Hans und Uwe Reyle (1993). From discourse to logic: introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory. Dordrecht: Kluwer.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä und Arto Anttila (Hg., 1995). Constraint Grammar. Berlin: de Gruyter.

Kohrt, Manfred. (1998a). Lexikalisch-semantische Eigenschaften von Fachsprachen. In: Hoffmann et al. (1998). S. 428-438

Kohrt, Manfred (1998b). Graphematische und phonologische Eigenschaften von Fachsprachen. In: Hoffmann et al. (1998). S. 438-442

Lamport, Leslie (1994). LATEX: a Document Preparation System. 2. Auflage. Reading, Massachusetts: Addison Wesley.

Lehrndorfer, Anne (1995). Kontrolliertes Deutsch: linguistische und sprachpsychologische Leitlinien für eine (maschinell) kontrollierte Sprache in der technischen Dokumentation. Tübingen: Narr.

Lie, Håkon Wium und Bert Bos (Hg., 1999). Cascading Style Sheets, level 1, W3C Recommendation 17 December 1996, revised 11 Jan 1999

Lobin, Henning (1993): Koordinationssyntax. Tübingen: Günther Narr Verlag.

Lobin, Henning (1999a, Hg.): Text im digitalen Medium. Linguistische Aspekte von Textdesign, Texttechnologie und Hypertext Engineering. Wiesbaden: Westdeutscher Verlag.

Lobin, Henning (1999b): Grammatische Restringierung von Dateninhalten in SGML/XML. In: Gippert (1999), S. 75-84.

Lobin, Henning (1999c): Intelligente Dokumente. Linguistische Repräsentation komplexer Inhalte für die hypermediale Wissensvermittlung. In: Lobin (1999a), S. 155-177.

Lobin, Henning (2000): Informationsmodellierung in XML und SGML. Berlin, Heidelberg: Springer-Verlag.

Lobin, Henning (Hg., 2001): Sprach- und Texttechnologie in digitalen Medien - Proceedings der GLDV-Frühjahrstagung 2001. Norderstedt: Books on Demand, 2001.

Lobin, Henning und Markus Reinsch (1999) Unification of XML Documents. In: InterChange. The Newsletter of The International SGML/XML Users' Group 5/2, S. 31-33.

Lobin, Henning und Andreas Witt (1999) Semantic and Thematic Navigation in Electronic Encyclopedias. In: Proc. of Electronic Publishing 99. Ronneby, S. 81-94.

Maler, Eve und Jean El Andaloussi (1996) SGML DTDs. From Text Developing to Model to Markup. Up-per Saddle River (NJ): Prentice Hall.

Mann, William C. und Sandra A. Thompson (1988): Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. In Text 8, S. 243-281.

McKelvie, David, Amy Isard, Andreas Mengel, Morten Baun M¢ller, Michael Grosse, Marion Klein (2001). The MATE workbench - An annotation tool for XML coded speech corpora. In: Speech Communication Vol. 33 (1-2), S. 97-112.

Megginson David (1998). Structuring XML documents. Upper Saddle River: Prentice Hall.

Mengel, A., L. Dybkjaer, J.M. Garrido, U. Heid, M. Klein, V. Pirrelli, M. Poesio, S. Quazza, A. Schiffrin und C. Soria (2000) MATE Dialog Annotation Guidelines (M-DAG). MATE Deliverable D2.1.

Mengel, Andreas und Ulrich Heid (1998). Cross Level Annotation. In: Bernsen et al. (1998). S. 55-65

Müller, St. (1999). Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche. Linguistische Arbeiten. Tübingen: Max Niemeyer Verlag.

Murata, Makoto, Dongwon Lee und Murali Mani (2001). Taxonomy of XML Schema Languages using Formal Language Theory, Proceedings of Extreme Markup Languages, Montreal, Canada, August, 2001.

Pollard, Carl und Ivan Sag (1994). Head-driven Phrase Structure Grammar. Chicago: The University of Chicago Press.

Pönninghaus, Jens (2000). Lexikalische Annotierung: Vom Stammformenlexikon zum annotierten Dokument mittels XML und DSSSL. Diplomarbeit an der Technischen Fakultät der Universität Bielefeld

Pönninghaus, Jens und Andreas Witt (2001). Lexikonexpansion: Vom XML-annotierten Stammformenlexikon zum Vollformenlexikon. In: Lobin (2001) S. 41-48

Pirelli, Vito und Claudia Soria (2000). Morphosyntax. In: Mengel et al. (2000), 44-117.

Rehm, Georg (1999): Automatische Textannotation. Ein SGML- und DSSSL-basierter Ansatz zur angewandten Textlinguistik. In: Lobin (1999a), 179-195.

Renear, Allen, Elli Mylonas und David Durand (1996). 'Refining Our Notion of What Text Really Is: The Problem of Overlapping Hierarchies' In: International Association for Literary and Linguistic Computing: Selected papers from the ALLC, ACH Conference: Christ Church, Oxford, April 1992. Oxford: Clarendon Press 1996.

Sailer, Manfred und Frank Richter (2001): Eine XML-Kodierung für AVM-Beschreibungen. In: Henning Lobin (Hg.) Tagungsband der GLDV-Tagung 2001. S. 161-168

Sasaki, Felix und Andreas Witt (2001): Präsentation, Transformation und Analyse: Verarbeitung XML-basierter japanischer Dialoge In: Lobin (2001) S. 169-178

Schachtl, Stephanie (1996). Requirements for Controlled German in Industrial Applications. In: Proceedings of the First International Workshop on Controlled Language Applications (CLAW 96), Leuven. S. 143-149

Schröder, Bernhard (1998): Pro-SGML: Ein Prolog-basiertes System zum Textretrieval. In: Heyer und Wolff (1998), S. 205-216.

Schwitter, Rolf (1998). Kontrolliertes Englisch für Anforderungsspezifikationen. Dissertationsschrift. Zürich: Studentendruckerei.

Simons, Gary F. und D. Terence Langendoen (1995). Rationale for the TEI Recommendations for Feature-Structure Markup. In: Ide und Véronis (Hg., 1995). S. 191 - 210.

Sperberg-McQueen, C. M. und Lou Burnard (Hg., 1994) Guidelines for Electronic Text Encoding and Inter-change (TEI P3). Chicago and Oxford: Text Encoding Initiative.

Sperberg-McQueen, C. M. und Claus Huitfeldt (1998). Concurrent Document Hierarchies in MECS and SGML, ALLC-ACH98, Joint Conference of the ALLC and ACH, Debrecen, Ungarn.

Sperberg-McQueen, C. M. und Claus Huitfeldt (1999). GODDAG: A Data Structure for Overlapping Hierarchies, ALLC-ACH99, Joint Conference of the ALLC and ACH, Charlottesville, Virginia.

Sperberg-McQueen, C. M., Claus Huitfeldt und Allen Renear (2000). Meaning and interpretation of Markup. In: Markup Languages: Theory & Practice 2.3, 215 - 234. MIT Press.

Thompson, Henry S., David Beech, Murray Maloney und Noah Mendelsohn (Hg., 2001). "XML Schema Part 1: Structures", W3C.

Uszkoreit, Hans (1986). Categorial unification grammars. In Proceedings of COLING 86, S. 187-194, Bonn.

Vlist, Eric van der (2001). "Examplotron", Internetdokument, März 2001.

Welty, Chris und Nancy Ide (1999). Using the right tools: enhancing retrieval from marked-up documents. In: Computers and the Humanities. 33(10). S. 59-84

Witt, Andreas (1996) Sprachverarbeitung mit getypten Attribut-Wert-Matrizen: Dependenzgrammatik und konzeptuelle Semantik. Magisterarbeit, Bielefeld.

Witt, Andreas (1998) TEI-based XML-Applications: Transcriptions. In: ALLC-ACH98, Joint Conference of the ALLC and ACH, Debrecen.

Witt, Andreas (1999a) SGML und Linguistik. In: Lobin (1999a). S. 121-153.

Witt, Andreas (1999b) DSSSL Zur Verarbeitung linguistischer Korpora. In: Gippert (Hg., 1999), S. 107-114.

Witt, Andreas und Harald Lüngen und Dafydd Gibbon (1997). Standardisierung orthographischer Transkriptionen: Ein SGML/TEI-basierter Vorschlag für VERBMOBIL, VERBMOBIL-Memo 117, Bielefeld.

Witt, Andreas, Harald Lüngen und Dafydd Gibbon (2000). Enhancing speech corpus resources with multiple lexical tag layers. Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000), Athen. S. 403-407

Anhang A: Architekturen

A.1 Die Basis- und die Meta-DTD aus Abschnitt 3.6

Basis-DTD

```
<?IS10744  ArcBase satzstruk>

<!ENTITY   SatzstrukturDTD SYSTEM "satzstruk.dtd">
<!NOTATION satzstruk PUBLIC
                 "-//LOCAL//NOTATION AFDR ARCBASE Book Architecture//EN">

<!ATTLIST #NOTATION satzstruk    ArcDTD    CDATA  "SatzstrukturDTD"
                                 ArcDocF   NAME   "titel"
                                 ArcFormA  NAME   "MetaStrukElement"
                                 ArcNamrA  NAME   "MetaStrukAttributs"
                                 ArcIgnDA  NAME   "Datenunterdruck">

<!ELEMENT title     (s)+>
<!-- Die Angabe der Architektonischen Beziehung                        -->
<!-- zwischen den Wurzelelementen: Ergibt sich bereits aus der Angabe -->
<!-- von ArcDocF, ist also redundant, aus Konsistenzgründen dennoch   -->
<!-- aufgeführt                                                       -->
<!ATTLIST title
              MetaStrukElement   NMTOKEN #FIXED   "titel"
              MetaStrukAttributs CDATA   "Inhalt #CONTENT">

<!ELEMENT s         (np|sp|vp|pp|punctuation)*>
<!ATTLIST s     type             CDATA   #IMPLIED
                MetaStrukElement   NMTOKEN #FIXED   "satzstruktur"
                MetaStrukAttributs CDATA   "typ type">

<!ELEMENT np         (w|sp|punctuation)+>
<!ATTLIST np     type             CDATA   #IMPLIED
                 MetaStrukElement   NMTOKEN #FIXED   "np"
                 MetaStrukAttributs CDATA   "typ type
                                     Inhalt #CONTENT">
<!ELEMENT sp         (#PCDATA)>
<!ELEMENT punctuation (#PCDATA)>
<!ELEMENT pp         (w|sp|np|punctuation)*>
<!ATTLIST pp
                 MetaStrukAttributs CDATA   "Inhalt #CONTENT">
<!ELEMENT w          (#PCDATA)>
```

Meta-DTD:

In der Basis-DTD wird auf sie mit dem Namen satzstruk.dtd verwiesen.

```
<!ELEMENT  titel        (satzstruktur+)>
<!ATTLIST  titel        Inhalt CDATA #IMPLIED>
<!ELEMENT  satzstruktur (np|pp|vp)*>
<!ATTLIST  satzstruktur typ CDATA #IMPLIED>
<!ELEMENT  np EMPTY>
<!ATTLIST  np typ     CDATA #IMPLIED
              Inhalt  CDATA #IMPLIED>
<!ELEMENT  pp EMPTY>
<!ATTLIST  pp
              Inhalt  CDATA #IMPLIED>
<!ELEMENT  vp EMPTY>
```

A.2 Beispiel der architektonischen Verarbeitung aus Abschnitt 6.2.3

In einem aus zwei Schritten bestehenden architektonischen Verarbeitung wird aus einem sehr reich annotierten Dokument ein sehr schwach annotierter Text abgeleitet.

Basis-DTD (Basis1)

```
<?IS10744  ArcBase word>
<!ENTITY   WordDocTyp        SYSTEM "wordDTDclient.dtd">
<!NOTATION word              PUBLIC
            "-//LOCAL//NOTATION AFDR ARCBASE Book Architecture//EN">
<!ATTLIST  #NOTATION word         ArcDTD    CDATA  "WordDocTyp"
                                  ArcDocF   NAME   "text"
                                  ArcFormA  NAME   "WordElements"
                                  ArcNamrA  NAME   "WordAttributes">
<!ELEMENT text
          (Nomen|Pronomen|Verb|Determinativ |
           Adjektiv|Präposition | Subjunktor |
           Konjunktor|Adverb|Modalpartikel|
           Rangierpartikel|Gradpartikel|Kopulapartikel|
           Satzäquivalente|unzuordenbar|
           Leerzeichen | Interpunktion)*>
<!ATTLIST text        WortElemente       NMTOKEN #FIXED   "text">
<!ELEMENT Leerzeichen      (#PCDATA)> <!ELEMENT Interpunktion    (#PCDATA)>
<!ELEMENT Nomen            (#PCDATA)> <!ELEMENT Pronomen         (#PCDATA)>
...
<!ELEMENT unzuordenbar     (#PCDATA)>
<!ATTLIST  Leerzeichen      WordElements   NMTOKEN #FIXED   "space">
<!ATTLIST  Interpunktion    WordElements   NMTOKEN #FIXED   "punctuation">
<!ATTLIST  Nomen            WordElements   NMTOKEN #FIXED   "word"
                            pos            CDATA #FIXED     "Nomen">
<!ATTLIST  Pronomen         WordElements   NMTOKEN #FIXED   "word"
                            pos CDATA #FIXED "Pronomen">
....
<!ATTLIST  unzuordenbar     WordElements   NMTOKEN #FIXED   "word"
                            pos            CDATA   #FIXED   "unzuordenbar">
```

Abgeleitetes Dokument

```
<!DOCTYPE text SYSTEM "wordDTDclient.dtd">
<text><word pos="Determinativ">Die</word><space> </space><word
pos="Adjektiv" >beiden</word><space> </space><word
pos="Nomen">Siebe</word><space> </space ><word
pos="Verb">kontrollieren</word><punctuation>.</punctuation></text>
```

Basis-DTD (Basis 2 ‚wordDTDclient.dtd', bereits Meta-DTD bzgl. Basis1):

```
<?IS10744  ArcBase wort>

<!ENTITY   WortDTD         SYSTEM "wortDTD.dtd">
<!NOTATION wort
           PUBLIC "-//LOCAL//NOTATION AFDR ARCBASE Book Architecture//EN">
<!ATTLIST  #NOTATION wort         ArcDTD    CDATA  "WortDTD"
                                  ArcDocF   NAME   "text"
                                  ArcFormA  NAME   "WortElemente"
                                  ArcNamrA  NAME   "WortDTDAttribute">
<!ELEMENT text     (word | space | punctuation | adj | n | det)*>
<!ATTLIST text
             WortElemente       NMTOKEN #FIXED   "text">

<!ELEMENT space       (#PCDATA)>
<!ELEMENT punctuation (#PCDATA)>
<!ELEMENT word        (#PCDATA)>
<!ATTLIST word        WortElemente   NMTOKEN #FIXED   "w"
                      pos            CDATA   #IMPLIED>
```

Abgeleitetes Dokument

```
<!DOCTYPE text SYSTEM "wortDTD.dtd">
<text><w>Die</w> <w>beiden</w> <w>Siebe</w> <w>kontrollieren</w>.</text>
```

Meta-DTD ‚wortDTD.dtd':

```
<!ELEMENT  text    (#PCDATA|w)*>
<!ELEMENT  w       (#PCDATA)>
```

Anhang B: Knoten im Dokument

Ausgabe der ESIS des annotierten Beispieltextes aus Abschnitt 5.2.1. In das Ausgabeformat wurden – abgesetzt durch ein Semikolon (;) die Knotenreferenzen eingearbeitet, die in der Prolog-Repräsentation des oben angeführten Beispielsatzes „Tauchen Sie das Gerät niemals in Wasser“ wiedergefunden werden können.

```
(text             ;Knoten 1
 (h1              ;Knoten 1 1
  (ell            ;Knoten 1 1 1
   -Reinigen      ;Datenknoten 1 1 1 1 - 1 1 1 8
   )ell)h1
 (h2              ;Knoten 1 2
  (ell            ;Knoten 1 2 1
   -Sandwichhalter reinigen:   ; Datenknoten 1 1 2 1 - 1 1 2 24
)ell )h2
 (ol              ;Knoten 1 3
  (li             ;Knoten 1 3 1
   Atype CDATA imp
   (s             ;Knoten 1 3 1 1
    -Entfernen Sie alle Reste von Brot und geschmolzenem Käse mit
einem hölzernen Spatel.         ; Datenknoten 1 3 1 1 1 ... 84
   )s  )li
  (li             ;Knoten 1 3 2
   Atype CDATA imp
   (s             ;Knoten 1 3 2 1
    -Spülen Sie den Sandwichhalter in heißem Wasser, dem Sie etwas
Spülmittel zugefügt haben, und trocken Sie ihn gründlich ab.
                  ; Datenknoten 1 3 2 1 1 ... 122
   )s
   Atype CDATA ind
   (s             ;Knoten 1 3 2 2
    -Sie können dieses Teil auch im Geschirrspüler reinigen.
                  ; Datenknoten 1 3 2 2 1 ... 55
    )s   )li  )ol
 (h2              ;Knoten 1 4
  (ell            ;Knoten 1 4 1
   -Den Toaster reinigen:     ; Datenknoten 1 4 1 1 ... 21
)ell )h2
 (ol              ;Knoten 1 5
  (li             ;Knoten 1 5 1
   Atype CDATA imp
   (s             ;Knoten 1 5 1 1
    -Ziehen Sie den Stecker aus der Steckdose.
                               ; Datenknoten 1 5 1 1 ... 41
)s  )li
  (li             ;Knoten 1 5 2
   Atype CDATA imp
   (s             ;Knoten 1 5 2 1
    -Reinigen Sie die Außenwände des Geräts mit einem feuchten
Tuch, auf das Sie bei Bedarf etwas Spülmittel aufgetragen haben.
                ; Datenknoten 1 5 2 1 ... 122
    )s
   Atype CDATA imp
   (s             ;Knoten 1 5 2 2
    -Sorgen Sie dafür, daß kein Wasser in das Gerät eindringt.
                  ; Datenknoten 1 5 2 2 1 ... 57
```

```
   )s
  Atype CDATA danger of life
  (warning     ;Knoten 1 5 2 3
   Atype CDATA imp
   (s          ;Knoten 1 5 2 3 1
    -Tauchen Sie das Gerät niemals in Wasser.
                ; Datenknoten 1 5 2 3 1 1 ... 40
    )s
   )warning
  )li
 (li           ;Knoten 1 5 3
  Atype CDATA imp
  (s           ;Knoten 1 5 3 1
   -Entfernen Sie Krümel nur, indem Sie die Krümelschublade ...
   )s
  Atype CDATA risk of damage
  (warning     ;Knoten 1 5 3 2
   Atype CDATA imp
   (s          ;Knoten 1 5 3 2 1
    -Schütteln Sie nicht die Krümel aus ...
    )s
   )warning
  )li
 (li           ;Knoten 1 5 4
  Atype CDATA ind
  (s           ;Knoten 1 5 4 1
   -Sie können das Netzkabel an der Unterseite ...
   )s
  )li
 )ol
)
```

Anhang C: Kontrollierte Sprache

C.1 DTD zur XML-Version des AECMA-Lexikons

```
<!ELEMENT Lexicon (entry*)>
<!ELEMENT entry   (approved | non-approved)>

<!ELEMENT word         (#PCDATA)>
<!ATTLIST word
                  id ID #IMPLIED>
<!ELEMENT approved      (word, pos, (meanings|ref))>
<!ELEMENT meanings      (meaning+, note*)>
<!ELEMENT meaning       (explanation, example, note*)>
<!ELEMENT explanation   (#PCDATA)>
<!ELEMENT example       (#PCDATA)>
<!ELEMENT note          (#PCDATA)>
<!ELEMENT pos           (#PCDATA)>
<!ELEMENT ref           EMPTY>
<!ATTLIST ref           reference IDREF #REQUIRED>

<!ELEMENT non-approved  (word, pos, approved-meanings)>

<!ELEMENT approved-meanings (approved-meaning+)>

<!ELEMENT approved-meaning ((approved-word | approved-expression), pos?,
approved-example, unapproved-example)>


<!ELEMENT approved-word (#PCDATA)>
<!ATTLIST approved-word reference IDREF #IMPLIED>

<!ELEMENT approved-expression (#PCDATA)>

<!ELEMENT approved-example     (#PCDATA)>
<!ELEMENT unapproved-example   (#PCDATA)>

<!--  verb (v) -->
<!--  noun (n) -->
<!--  adjective (adj) -->
<!--  adverb (adv) -->
<!--  article (art) -->
<!--  preposition (pre) -->
<!--  pronoun (pn) -->
<!--  conjunction (con) -->

<!ATTLIST pos cat      (verb|noun|adjective|adverb|article|preposition|
                        pronoun|conjunction|prefix)              #IMPLIED
              vform    (vform1|vform2|vform3|vform4)             #IMPLIED
              aform    (basic|comparative|superlative)           #IMPLIED
>
<!--  vform1:  infinitiv            (to adjust ... oder ... adjust ...) -->
<!--           imperativ            (Adjust ...)           -->
<!--           simple present tense (ausser 3.Person sg.)  -->
<!--           future tense         (you will ADJUST)      -->
<!--  vform2:  simple present tense 3.Person sg.           -->
<!--  vform3:  simple past tense    (it ADJUSTED)          -->
<!--  vform4:  Partizip             (past participle)      -->
<!--  aform:   Steigerungsform -->
```

C.2 XML-Version des AECMA-Lexikons

Die XML-Version des im Abschnitt 7.2.2.1 abgebildeten Ausschnitts des Lexikons ist nachfolgen dargestellt.

```
<!DOCTYPE  Lexicon SYSTEM "lexikon.dtd">
<Lexicon>
...
  <entry>
    <approved>
      <word id="incident_n1">incident</word>
      <pos cat="noun">noun</pos>
      <meanings>
         <meaning>
           <explanation> An important "occurrence" that can cause damage or
have dangerous results</explanation>
           <example> RECORD ALL INCIDENTS OF WATER FOUND IN THE
FUEL.</example>
         </meaning>
      </meanings>
    </approved>
  </entry>
  <entry>
    <non-approved>
      <word id="incline_n1">incline</word>
      <pos cat="noun">noun</pos>
      <approved-meanings>
         <approved-meaning>
           <approved-word reference="slope_n1">slope</approved-word>
           <pos cat="noun">noun</pos>
           <approved-example> IF YOU MUST TOW THE AIRCRAFT DOWN A SLOPE,
THERE MUST BE A PERSON IN THE COCKPIT TO OPERATE THE BRAKE IF
NECESSARY.</approved-example>
           <unapproved-example> If the aircraft has to be towed down an
incline, there must be someone in the cockpit to operate the brake if
necessary.</unapproved-example>
         </approved-meaning>
      </approved-meanings>
    </non-approved>
  </entry>
  <entry>
    <approved>
      <word id="include_v1">include</word>
      <pos cat="verb">verb</pos>
      <meanings>
         <meaning>
           <explanation> To make, or to be, part of</explanation>
           <example> THIS CHAPTER INCLUDES THE PROCEDURES FOR THE REMOVAL OF
THE LANDING GEAR.</example>
         </meaning>
      </meanings>
    </approved>
  </entry>
  <entry>
    <approved>
      <word>includes</word>
      <pos cat="verb" vform="vform1">verb</pos>
      <ref reference="include_v1"/>
```

```
    </approved>
  </entry>
  <entry>
    <approved>
      <word>included</word>
      <pos cat="verb" vform="vform2">verb</pos>
      <ref reference="include_v1"/>
    </approved>
  </entry>
  <entry>
    <approved>
      <word>included</word>
      <pos cat="verb" vform="vform3">verb</pos>
      <ref reference="include_v1"/>
    </approved>
  </entry>
  <entry>
    <non-approved>
      <word id="including_pre1">including</word>
      <pos cat="preposition">preposition</pos>
      <approved-meanings>
         <approved-meaning>
           <approved-word reference="thru_pre1">thru</approved-word>
           <pos cat="preposition">preposition</pos>
           <approved-example> DO TESTS 4 THRU 8 AGAIN.</approved-example>
           <unapproved-example> Repeat from test 4 up to and including test
8.</unapproved-example>
         </approved-meaning>
      </approved-meanings>
    </non-approved>
  </entry>
  <entry>
    <non-approved>
      <word id="including_pre2">including</word>
      <pos cat="preposition">preposition</pos>
      <approved-meanings>
         <approved-meaning>
           <approved-word reference="with_pre1">with</approved-word>
           <pos cat="preposition">preposition</pos>
           <approved-example> RETURN THE DEFECTIVE COVER, WITH THE OIL
SAMPLES, TO THE REPAIR CENTER.</approved-example>
           <unapproved-example> Return defective cover, including oil
samples, to the repair center.</unapproved-example>
         </approved-meaning>
      </approved-meanings>
    </non-approved>
  </entry>
...
</Lexicon>
```

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbstständig verfasst, keine anderen als die angegebenen Hilfsmittel verwendet und wörtlich oder inhaltlich übernommene Stellen als solche gekennzeichnet habe.

Gedruckt auf alterungsbeständigem Papier ISO ∞ 9706

Bielefeld, den 25. November 2001 Andreas Witt