Goecke, Daniela/Naber, Daniel/Witt, Andreas (2003): Query von Multiebenen-annotierten XML-Dokumenten mit Prolog. In: Seewald-Heeg, Uta (Hrsg.): Sprachtechnologie für die multilinguale Kommunikation. Textproduktion, Recherche, Übersetzung, Lokalisierung. Beiträge der GLDV-Frühjahrstagung 2003. Sankt Augustin, 391–405. (Sprachwissenschaft, Computerlinguistik und Neue Medien - 5).

Daniela Goecke, Daniel Naber, Andreas Witt

Query von Multiebenen-annotierten XML-Dokumenten mit Prolog

1 Einleitung

In dem Beitrag wird ein Ansatz vorgestellt, XML-Dokumente zu analysieren, die hinsichtlich mehrerer Ebenen annotiert sind. Die Arbeiten stehen in Zusammenhang mit einem Projekt, in dem sprachliche Funktionen (insbesondere Koreferenz) in Beziehung gesetzt werden zu den sprachlichen Ausdrucksmitteln der untersuchten typologisch unterschiedlichen Sprachen[1]. Derartige Untersuchungen bilden für eine multilingual ausgerichtete Sprachtechnologie eine wichtige Grundlage. So ist es z.B. für maschinelle Übersetzungen unabdingbar, Kenntnisse über eine angemessene Versprachlichung einer in einer Quellsprache als Pronomen realisierten koreferenten Einheit zu besitzen.

Als Beispiel für die Relevanz der Verknüpfung von verschiedenen Ebenen soll in dem vorliegenden Beitrag jedoch eine andere Anwendung der hier vorgestellten Query-Möglichkeiten beschrieben werden, die ebenfalls für Arbeiten im Bereich der maschinellen Übersetzung sehr relevant ist, nämlich die Kontrollierte Sprache (vgl. Lehrndorfer 1996).

Bei der Analyse von XML-Dokumenten liegt unser Schwerpunkt auf dem Vergleich von Annotationsebenen, existierende Query-Sprachen (vgl. Bonifati & Lee, 2001) legen jedoch zumeist den Schwerpunkt auf die Analyse einer einzelnen Annotationsebene. In Sprachen wie XQuery 1.0 oder XPath 1.0 werden Anfragen in XML-Syntax formuliert. Das Datenmodell ist eine Baumstruktur, Inklusionsrelationen zwischen Elementen, die durch die hierarchische Struktur der Daten gegeben sind, können einfach erfragt werden. Die Query-Sprachen erlauben zwar die Verknüpfung von verteilten Annotationen, überlappende Elemente lassen sich jedoch nicht darstellen, da für deren Modellierung eine parallele Sicht auf die Primärdaten notwendig ist.

Um eine parallele Sicht zu ermöglichen, wird das Datenmodell um zusätzliche Informationen erweitert, die eine Verknüpfung der verschiedenen Annotationsebenen erlauben. Für die Realisierung dieses Ansatzes wird der Inferenzmechanismus der Programmiersprache Prolog verwendet[2].

---

[1] Im Projekt A2 der DFG Forschergruppe 437 „Texttechnologische Informationsmodellierung" wird Koreferenz als sprachliche Funktion untersucht hinsichtlich ihrer sprachlichen Realisierungen in verschiedenen Sprachen (Japanisch, Kilivila, Deutsch).

[2] Die Verwendung von Prolog für Analysen von Multi-Ebenen-Daten hat zudem den Vorteil, dass existierende computerlinguistische Werkzeuge, die wie im VERBMOBIL-Verbund- projekt häufig in Prolog realisiert sind, für XML-annotierte Daten wiederverwendet werden können.

Ein prologbasiertes System zum Textretrieval für SGML-Daten stellt Schröder (1998) vor. Ein neuerer Ansatz, der Prolog für XML-Annotationen verwendet, findet sich in Arbeiten von Sperberg-McQueen et al. (2001, 2002). Die Überführung der XML-Annotationen in eine Prolog-Faktenbasis bietet die Möglichkeit, alle Arten von Anfragen zu realisieren, die sich in Hornklausellogik, einer Untermenge der Prädikatenlogik erster Stufe, beschreiben lassen.

Auf die speziellen Eigenschaften von Multi-Ebenen-Annotationen und der Anwendung „Kontrolliert Sprache" wird im nächsten Abschnitt eingegangen; die Implementierung des Ansatzes wird in Abschnitt 3 dargestellt und anhand von Beispielen illustriert. Der Beitrag schließt mit einem Ausblick in Kapitel 4.

2 Multi-Ebenen-Annotationen

Im Allgemeinen wird davon ausgegangen, dass die Informationen, die einem Dokument hinzugefügt werden, hierarchischer Natur sind, und dass die einzelnen informationellen Einheiten kompatibel sind. Diese Auffassung, dass es sich bei Texten um eine „ordered hierarchy of content objects" handelt wird als OCHO-These bezeichnet. Insbesondere für eine linguistische Analyse von Sprachdaten – aber auch für formale Textsorten wie z.B. technische Dokumentationen – lässt sich diese Sichtweise jedoch nicht aufrechterhalten (vgl. Simons 1998). Eine kritische Diskussion der OCHO-These findet sich bei Renear et al. (1996).

Beispiele für Beschreibungsebenen, in denen Elemente häufig nicht hierarchisch organisiert werden können, sind die Absatz- und Seitenstruktur oder die Morphem- und Silbenstruktur. Auch die Ebenen Syntax und sprachliche Funktion lassen sich oft nicht hierarchisch zueinander in Beziehung setzen. Elemente der genannten Ebenen können sich überlappen oder wechselseitig einschließen. Die Verteilung von Informationen auf mehrere Ebenen kann aber auch als eine Fokussierung der Informationsmodellierung aufgefasst werden, selbst dann, wenn eine Hierarchie gefunden werden könnte, in der die verschiedenen Ebenen enthalten sind.

Darüber hinaus ist eine Trennung der Auszeichnungsebenen dann sinnvoll, wenn kein oder nur wenig Wissen darüber besteht, wie die gemeinsame Ebene z.B. als eine Dokumentgrammatik definiert werden kann. Dieses Wissen kann durch die Analyse getrennt annotierter Ebenen aus den Daten extrahiert werden.

2.1 Verknüpfung multipler Annotationen

Wie in der Einleitung dargestellt wurde, ist für eine Modellierung von sich überlappenden Einheiten eine parallele Sicht auf die Daten notwendig. Um eine parallele Sicht zu ermöglichen, wird das Datenmodell der XML Query-Sprachen erweitert um Informationen über die Position der ausgezeichneten Elemente innerhalb der Primärdaten. Dies erlaubt die Bezugnahme auf eine absolute Referenzebene, nämlich die Daten selbst. Primärdaten sind diejenigen Daten, die den Annotationen zugrunde liegen, d.h. die noch nicht annotierten Daten. Die verschiedenen Annotionsebenen sind Metadaten zu den Primärdaten (vgl. Witt 2002).

Für einen Vergleich verschiedener Ebenen kann somit auf ausgezeichnete Textabschnitte zugegriffen werden. Dementsprechend können einzelne **Elementinstanzen** von in der Dokumentgrammatik definierten Elementen verglichen werden, oder es können Aussagen über die gesamte **Elementklasse** getroffen werden, d.h. über alle Instanzen eines Elements.

2.2 Beschreibungsebenen für Kontrollierte Sprachen

Für alle Beispiele wird ein Ausschnitt aus der Domäne Kontrollierte Sprachen verwendet, da hier oftmals eine Modellierung mehrerer Ebenen notwendig ist. Kontrollierte Sprachen definieren bestimmte Restriktionen, um beispielsweise technische Dokumentationen einheitlich und verständlich zu gestalten oder Ambiguitäten auszuschließen, die eine maschinelle Übersetzung erschweren würden. Diese Restriktionen können auf verschiedenen Ebenen angesiedelt sein (z.B. Semantik, Syntax, Interpunktion, Layout). Die Verbindung dieser Ebenen ist Gegenstand aktueller Diskussionen, so bezeichnen Hartley und Paris (2001) die (fehlende) Schnittstelle zwischen Layout-Spezifikationen und den sprachbezogenen Regeln als „the controlled language gap". Für die Überprüfung der nachstehenden editierspezifischen Regel muss sowohl die syntaktische als auch die textuelle Struktur modelliert werden[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 Satz, d.h. einen ganzen Satz oder mehrere Sätze.

Der folgende Abschnitt dient als Primärtext, die Zeilennummern wurden hinzugefügt, um eine Referenz im Text zu ermöglichen[4].

Anmerkungen (Z1)

---

[3] Die Regel wird beispielsweise in einer Empfehlung für Fachtexte der Siemens Nixdorf Informationsysteme AG ausführlich beschrieben (vgl. auch Schachtl 1996).

[4] Bedienungsanleitung eines Sony Mini-Disc Recorders.

- Wenn keine CD für die gewählte Discnummer eingelegt ist, erscheint „No Disc“ im Display. (Z2) (Z3)
- Der Mega CD Control-Modus kann aktiviert werden, ohne Rücksicht darauf, ob sich eine MD im MD-Deck befindet oder nicht. (Z4) (Z5)
- Der Mega CD Control-Modus wird deaktiviert, wenn: (Z6)
  - das MD-Deck ausgeschaltet wird. (Z7)
  - die MD ausgeworfen wird. (Z8)

Der Beispieltext verstößt gegen die Regel, da die beiden eingeschachtelten Listenelemente (Z7,Z8) keine syntaktisch korrekten Sätze bilden und nur das erste der beiden Elemente mit dem Kontext gemeinsam einen vollständigen Satz bildet[5].

Die Primärdaten werden hinsichtlich zweier Ebenen annotiert: HTML zur Darstellung der textuellen Struktur und eine Annotierung der syntaktischen Information. Die Dokumentgrammatik für die zweite Ebene definiert die Elemente „sentence“ für vollständige Sätze und „text“ für unvollständige Sätze. Über einen Vergleich dieser Ebenen lassen sich Aussagen darüber treffen, welche Relationen zwischen dem Element „sentence“ und dem Listenelement „LI“ der HTML-Ebene gelten und ob diese Relationen der oben genannten Regel entsprechen.

3 Implementierung

In dem Datenmodell des hier vorgestellten Ansatzes wird zusätzlich zu der DOM-Baumstruktur der XML-Annotationen eine sequentielle Struktur der annotierten Textsegmente modelliert, um überlappende Elemente beschreiben zu können.

Die XML-Annotationen der Primärdaten werden zunächst durch ein Python-Skript in Prolog-Fakten übersetzt. Die Prologfakten, in denen sowohl die Baumstruktur als auch die sequentielle Struktur kodiert ist, bilden die Eingabe für das Prolog-Programm. Im Folgenden werden beide Bereiche dargestellt.

3.1 Python

Die Umwandlung von XML-Dateien in ein Prolog-Programm wird von einem Python-Skript[6] übernommen, das rekursiv über den DOM-Baum der XML-Dateien iteriert.

---

[5] Eine der Regel entsprechende Version ist z.B. der Satz „Der Mega CD Control-Modus wird deaktiviert, wenn das MD-Deck ausgeschaltet wird oder die MD ausgeworfen wird.“ Eine andere regelkonforme Paraphrasierung wäre: „Der Mega CD Control-Modus wird in den folgenden Fällen deaktiviert: - Das MD-Deck wird ausgeschaltet. - Die MD wird ausgeworfen.“

Als Eingabe können einzelne oder beliebig viele XML-Dateien benutzt werden. Ausgegeben wird eine Prolog-Datei, die dann in die interaktive Prolog-Umgebung importiert werden kann. XML-Elemente werden in ein 5-stelliges Prolog-Fakt node/5 übersetzt, für Attribute wird ein 6-stelliges Prolog-Fakt attr/6 generiert.

node(AnnotationLayer, Start, End, Node, element(ElementName)).
- AnnotationLayer ist die Annotationsebene.
- Start ist die Startposition im Primärtext.
- End ist die Endposition im Primärtext.
- Node beschreibt den Knoten im DOM-Baum.
- element(ElementName) ist der Name des Elements.

attr(AnnotationLayer, Start, End, Node, AttributeName, Value).
- AnnotationLayer, Start, End, sowie Node sind wie für node/5 definiert.
- AttributeName den Namen des Attributs beschreibt, und
- Value der Wert des Elements ist.

Der Aufruf des Programms für die beiden Beschreibungsebenen der editierspezifischen Regel für Bedienungsanleitungen ergibt die Ausgabe in Abbildung 1.
Zusätzliche Parameter können die Ausgabe beeinflussen[7]:

- Es kann geprüft werden, ob alle Eingabedateien auf den gleichen Primärdaten aufbauen. Sofern in den XML-Eingabedateien eine DTD angegeben ist, werden die Dateien bei der Verarbeitung automatisch validiert. Dabei wird Whitespace, der laut DTD nicht relevant ist, ignoriert. Sind die Primärdaten nicht identisch, werden die Position und der Kontext der ersten abweichenden Position ausgegeben. Wird keine DTD angegeben findet keine Validierung statt und jeglicher Whitespace ist relevant.
- Es können Prolog-Fakten für jeden Buchstaben aus den Primärdaten generiert werden: **node('sentence.xml',0,1,[1,1,1],pcdata('A')).**
- Für jeden Buchstaben kann ein eigener Prolog-Fakt unabhängig von der Beschreibungsebene erzeugt werden: **pcdata_node(0,1,'A')**. Die ersten beiden Argumente geben die Start- bzw. Endposition an und das dritte Argument ist der Buchstabe aus den Primärdaten.

---

[6] Als Implementierungssprache wurde Python gewählt, da es aufgrund seiner guten XML- und Unicode-Unterstützung für diese Aufgabe optimal geeignet ist. Durch die einfache Syntax und die umfangreiche mitgelieferte Standardbibliothek ist eine schnelle Softwareentwicklung möglich. Das Skript und eine Dokumentation dazu kann heruntergeladen werden unter http://coli.lili.uni-bielefeld.de/Texttechnologie/Forschergruppe/prolog/.

[7] Aufruf: xml2prolog.py [-h|--help] [-c|--compare] [-p|--pcdata] [-n|--pcdatanodes] <files...>

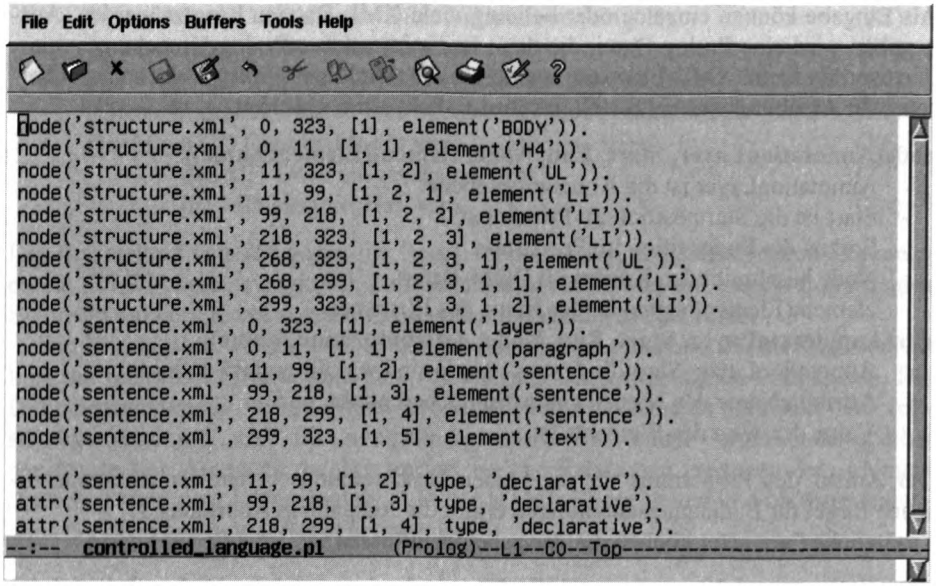

Abb. 1: Prolog-Faktenbasis.

3.2 Prolog

Als Prolog-Implementierung wurde SWI-Prolog (Version 5.0.1), ein frei verfügbarer Prolog-Compiler, gewählt. Zum jetzigen Zeitpunkt werden alle Anfragen in der Prolog-Eingabeaufforderung formuliert. Es ist jedoch auch möglich, die Prädikate von anderen Modulen aus aufzurufen, die in anderen Programmiersprachen realisiert sind. Für SWI-Prolog existieren Schnittstellen zu den Programmiersprachen C, C++ oder Java.

Die Funktionalität umfasst drei Bereiche, die z.T. unterschiedlichen Zielen dienen.

- Statistiken über das Annotationsinventar
- Informationen über Elementinstanzen
- Informationen über Elementklassen

Die Prädikate zu diesen Bereichen werden im nachfolgenden Abschnitt anhand von Beispielen erläutert. Als Eingabedaten wird die Prologfaktenbasis verwendet, die mit dem Python-Skript für die Beispieldaten der editierspezifischen Regel generiert wurde.

3.2.1 Statistik über Beschreibungsebenen

Die Erstellung einer Statistik dient einem ersten Überblick über die annotierten Daten und erlaubt es somit auch Personen, die bisher nicht mit dem Annotationsinventar gearbeitet haben und keine Informationen über die Struktur der Daten haben, die Daten zu interpretieren. Die Ausgabe der Statistik kann entweder in eine Datei geschrieben werden oder die Statistik wird – wie in Abb. 2 – interaktiv gestartet.

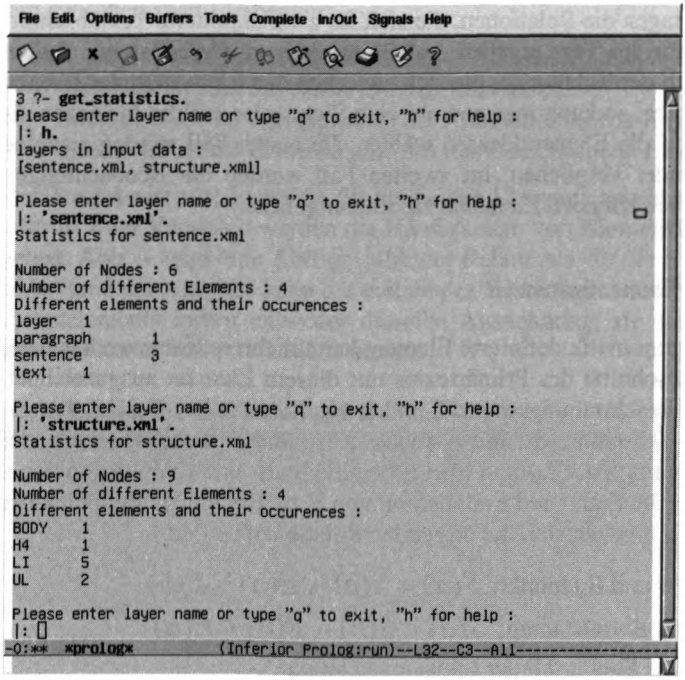

Abb. 2: Interaktive Statistik über Annotationsebenen.

Die Ausgabe zeigt, dass in den Primärdaten drei Textabschnitte als Satz (sentence) annotiert sind und dass ein Textabschnitt keinen vollständigen Satz bildet (text). Hinsichtlich der strukturellen Ebene, sind die Primärdaten als fünf Listenelemente (LI) modelliert, die in einer verschachtelten Liste enthalten sind.

3.2.2 Vergleich von Annotationsebenen

Weitere Prädikate dienen dem Vergleich verschiedener Annotationsebenen miteinander. Die zugrundeliegenden Primärdaten, d.h. die Menge aller Daten, die als #PCDATA definiert sind, dienen als verknüpfendes Element der auf verschiedenen Ebenen annotierten Textversionen (vgl. Witt 2002). Vergleiche der einzelnen Annotationsebenen bauen auf der Position der annotierten Textelemente im Primärtext auf. Jeder Textabschnitt ist beschrieben durch seine Start- und Endposition, Duruseau & Brook O'Donnell (2002) übertragen die Relationen, die sich für zwei annotierte Textabschnitte aufgrund ihrer Position im Text ergeben, auf Elemente von XML-Dateien. Sollen zwei Ebenen hinsichtlich der Relationen, die sich zwischen den Elementen der Ebenen ergeben, analysiert werden, so kann man sowohl eine Sichtweise auf die Instanzen als auch eine Sichtweise auf die Elementklassen wählen: Im ersten Fall werden einzelne Textabschnitte miteinander verglichen, im zweiten Fall werden die Beobachtungen über alle Instanzen der betreffenden Elemente zusammengefasst.

3.2.2.1 Vergleich von Elementinstanzen

Jedes in der Dokumentgrammatik definierte Element kann mehrere Instanzen besitzen, d.h. es sind mehrere Abschnitte des Primärtextes mit diesem Element ausgezeichnet. Sei A ein Element der Beschreibungsebene 1 und B ein Element der Beschreibungsebene 2, so ist a ein Textabschnitt, der mit A annotiert ist, und b ist ein Abschnitt, der mit dem Element B annotiert ist. S(a) und E(a) geben die Start- und Endposition von a an, S(b) und E(b) geben die Start- und Endposition von b an. Bei einem Vergleich der Start- und Endpositionen ergeben sich die folgenden Relationen:

- **Identität** von Start- und Endpunkt: $S(a) = S(b) \wedge E(a) = E(b)$
- **Inklusion** A enthält B vollständig: $S(a) < S(b) \wedge E(b) < E(a)$
- **Endpunkt-Identität** Sonderfall der Inklusionsrelation:
  $$S(a) < S(b) \wedge E(a) = E(b)$$
- **Startpunkt-Identität** Sonderfall der Inklusionsrelation:
  $$S(a) = S(b) \wedge E(a) < E(b)$$
- **Endpunkt=Startpunkt**: $E(a) = S(b)$
- **Überlappung**: $(S(a) < S(b) < E(a)) \wedge (E(b) > E(a))$
- **Unabhängigkeit**: $(E(a) < S(b)) \vee (E(b) < S(a)$

Für zwei Instanzen a von A und b von B ergeben sich insgesamt 15 mögliche Relationen, abhängig davon, ob im Primärtext a vor b oder b vor a steht bzw. ob im Primär-

text a oder b die größere Ausdehnung hat. Die Inklusionsrelation kann, wie oben beschrieben durch S(a) < S(b) und E(b) < E(a) gegeben sein, es kann aber auch gelten, dass S(b) < S(a) und E(a) < E(b).

3.2.2.2 Prädikate

Das Prädikat **chk_relation/6** erlaubt die Berechnung von Informationen über Relationen zwischen zwei Elementen:

chk_relation(Relation,Element1,Layer1,Element2,Layer2,L).

- **Relation** ist eine der beschriebenen Relationen,
- **Element1** ist der Elementname der Annotationsebene **Layer1** und
- **Element2** ist der Elementname aus Annotationsebene **Layer2**.

Die Ausgabe ist eine Liste, die für die beiden Elemente angibt, wie häufig die Relation auftritt. Zum Vergleich werden die Häufigkeiten von **Element1** und **Element2** ausgegeben. Abb. 3 zeigt eine Abfrage aller der Relationen für die Elemente LI und sentence, anhand der Ausgabe kann die editierspezifische Regel überprüft werden: Sätze und Listenelemente haben entweder dieselbe Ausdehnung, sie sind also identisch, oder Sätze umfassen mehrere Listenelemente.

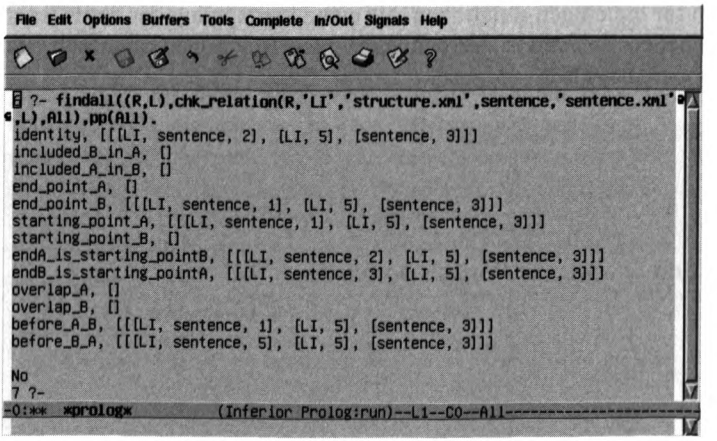

Abb. 3: Mögliche Relationen zwischen den Elementen LI und sentence.

In zwei Fällen sind die Elemente identisch, die betreffenden Textabschnitte entsprechen somit der editierspezifischen Regel. In einem Fall (end_point_B) schließt eine Instanz von „sentence" eine Instanz von „LI" ein und in einem weiteren Fall (star-

ting_point_A) schließt „LI“ „sentence“ ein. Die weiteren Relationen sind für eine Überprüfung der Regel nicht von Interesse, da sie voneinander unabhängige Textabschnitte beschreiben: endA_is_starting_pointB und endB_is_starting_point_A beschreiben diejenigen Fälle, in denen die Relationen direkt aneinander anschließen, before_A_B und before_B_A beschreiben solche Textabschnitte, die nicht aneinander anstoßen (siehe auch Abb. 5). Die Knoten [1,2] (Z2 und Z3 im obigen Beispieltext) und [1,3] (Z4 und Z5) der Ebene sentence.xml stehen vor den Knoten [1,2,3,1,1] (Z7) und [1,2,3,1,2] (Z8) der Ebene structure.xml. Knoten [1,2] steht zusätzlich noch vor Knoten [1,2,3] (Z6-Z8).

Die Information darüber, für welche Elementinstanzen eine Relation zutrifft, kann dem Prädikat **chk_relation_el/8** abgefragt werden:

chk_relation_el(Rel,E1,Layer1,E2,Layer2,L,CounterEx1, CounterEx2)

- Rel ist die Relation,
- E1 ist der Elementname der Annotationsebene Layer1,
- E2 ist der Elementname aus Annotationsebene Layer2,
- L ist die Liste aller Vorkommen von Relation Rel,
- CounterEx1 ist die Liste der Gegenbeispiele in Layer1, also die Liste aller Vorkommen von E1, für die Rel nicht gilt, und
- CounterEx2 ist die Liste der Gegenbeispiele in Layer1, also die Liste aller Vorkommen von E1, für die Rel nicht gilt.

Abb. 4 zeigt Anfragen für diejenigen Relationen, die eine Überlappung der betroffenen Textabschnitte einschließen. In Abb. 5 werden diejenigen Relationen abgefragt, die unabhängige Textelemente betreffen.

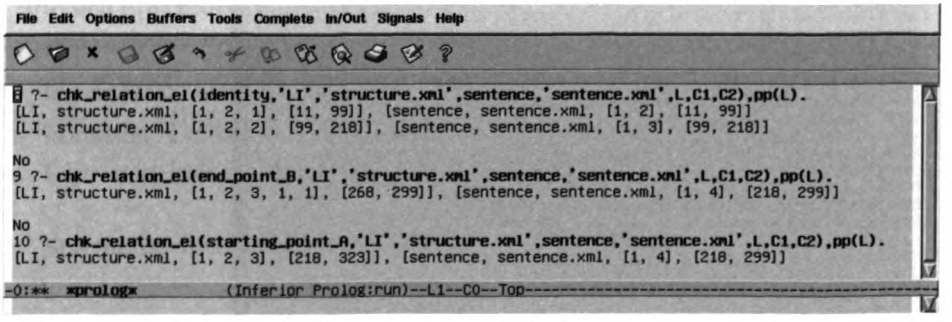

Abb. 4: Informationen über Elementinstanzen (überlappende Relationen).

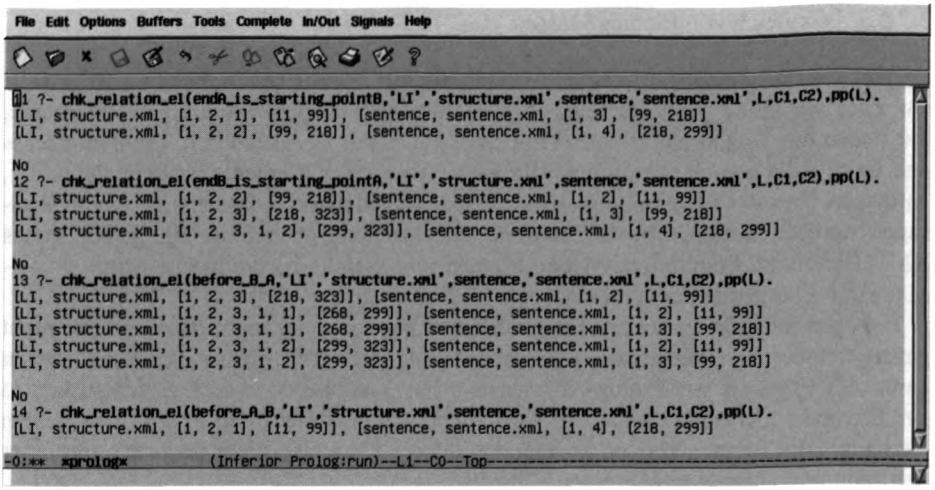

Abb. 5: Informationen über Elementinstanzen (unabhängige Relationen).

Bezogen auf die Regel, dass jedes Listenelement entweder einen vollständigen Satz bilden muss, oder gemeinsam mit dem Kontext einen Satz bilden muss, zeigt die erste Abfrage in Abb. 4, dass die Abschnitte Z2-Z3 (Startposition unmittelbar nach Position 11, Ende mit der Position 99) und Z4-Z5 (99,218) der Primärdaten dieser Regel entsprechen. Die folgende Abfrage zeigt, dass ein Listenelement von Z7 (Knoten [1,2,3,1,1]) in dem Satz von Z6-Z7 (Knoten [1,4]) enthalten ist (Relation end_point_B), dies entspricht der editierspezifischen Regel, da das Listenelement mit dem Kontext einen vollständigen Satz bildet. Der Satz selbst steht wiederum zu Beginn des Listenelements Z6-Z8 (Knoten [1,2,3], Relation starting_point_A). Abschnitte in den Primärdaten, die dieser Regel nicht entsprechen, d.h. solche Daten, die keinen vollständigen Satz bilden (Z8), wurden mit dem Element „text" ausgezeichnet, und können – wie in Abb. 6 – ebenfalls einfach abgefragt werden.

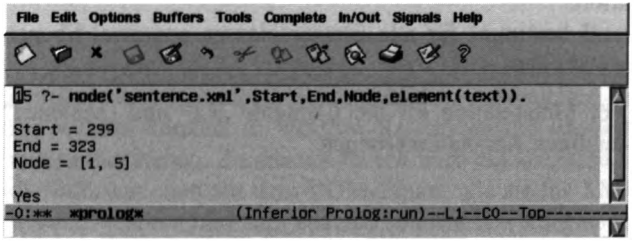

Abb. 6: Anfrage für alle Primärdatenabschnitte, die mit „text" annotiert sind.

3.2.2.3 Vergleich von Elementklassen

Die im vorangegangenen Abschnitt beschriebenen Relationen beziehen sich jeweils auf zwei annotierte Textabschnitte, d.h. es handelt sich um Instanzen der Elemente, mit denen die Textabschnitte annotiert sind. Für einen Vergleich der Annotationsebenen ist es jedoch auch notwendig, alle Instanzen zweier Elemente zu betrachten, d.h. Aussagen über die Elementklassen zu treffen. Beispielsweise lassen sich nur so Aussagen darüber treffen, ob verschiedene Beschreibungsebenen in ein gemeinsames XML-Dokument kombiniert werden können und welche hierarchische Struktur zwischen den Elementen der verschiedenen Ebenen besteht.

Bezogen auf alle Instanzen zweier Elementklassen zeigt sich, dass zumeist nicht genau eine der im vorangegangen Abschnitt genannten Relationen zugeordnet werden kann. Als Ergebnis der Prolog-Abfrage in Abb. 3 existieren zwischen den Instanzen der Elemente „LI“ und „sentence“ verschiedene Relationen. Die Frage, in welchem Verhältnis die Elementklassen zueinander stehen, kann also nicht eindeutig mit einer der Relationen beantwortet werden.

Zu diesem Zweck wurden Metarelationen definiert, die einen Vergleich von Elementklassen erlauben. Diese Metarelationen fassen die Sonderfälle der Inklusionsrelation zusammen, so dass sich vier Metarelationen ergeben:

- **Unabhängigkeit** Für alle Instanzen a und b der Klassen A und B gelten ausschließlich die Relationen Unabhängigkeit und Endpunkt=Startpunkt.
- **Identität** Für alle Instanzen von A und B, die nicht unabhängig voneinander sind, gilt ausschließlich die Identitätsrelation.
- **Inklusion** Für alle Instanzen von A und B, die nicht identisch oder unabhängig voneinander sind, gelten ausschließlich die Relationen Inklusion, Startpunkt-Identität und Endpunkt-Identität.
- **Überlappung** Für alle nicht-unabhängigen Instanzen von A und B existiert nur die Überlappungs-Relation.

Für eine Analyse der Metarelationen stehen zwei Prädikate zur Verfügung:

- **chk_metarelation_el/5** bestimmt für zwei Elementklassen aus zwei Ebenen die entsprechende Metarelation.
- **chk_metarelation_layer/3** bestimmt für alle Elementklassen aus zwei Ebenen die entsprechende Metarelationen.

Abb. 7 zeigt eine Abfrage der Metarelation für die Elemente „LI“ und „sentence“ sowie eine Analyse der vollständigen Annotationsebenen.

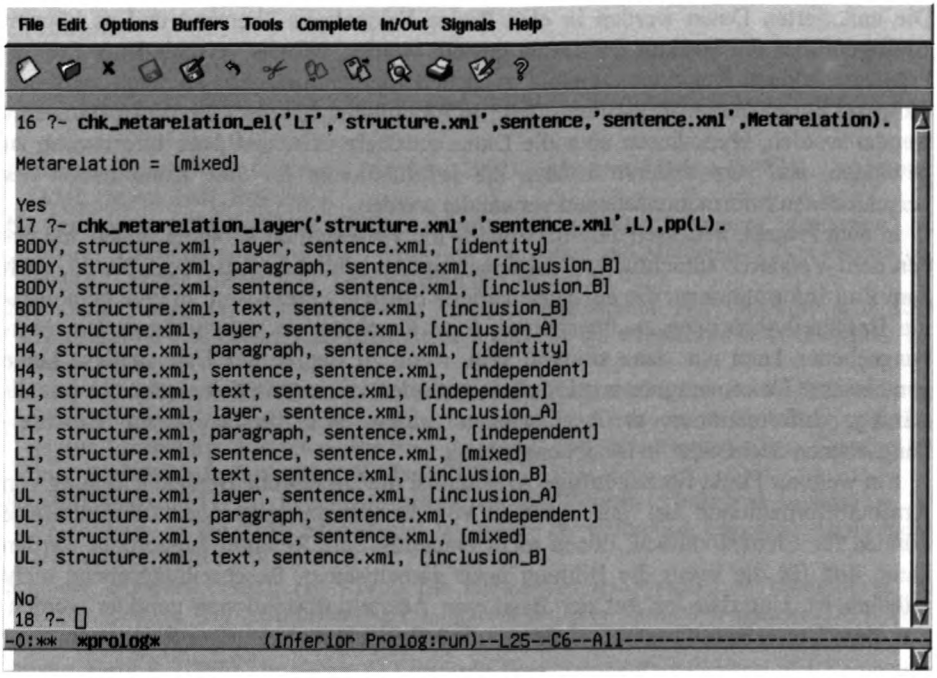

Abb. 7: Metarelationen für zwei Elemente und für Annotationsebenen.

Die Metarelation für die Elemente sentence und LI ist „mixed", da für einige Instanzen gilt, dass sentence LI enthält, und für andere Instanzen gilt, dass LI sentence beinhaltet (die Relationen end_point_B und starting_point_A, s. Abb. 4).

4 Zusammenfassung und Ausblick

Es wurde ein Ansatz vorgestellt, verschiedene XML-Dokumente, welche die selben Primärdaten beschreiben, dahingehend zu untersuchen, dass Aussagen darüber getroffen werden können, in welchen Relationen die Elemente der Beschreibungsebenen zueinander stehen. Zu diesem Zweck wird ein Datenmodell verwendet, das zum einen die Informationen aus dem DOM-Baum, wie sie für XML Query-Sprachen verwendet werden, umfasst und zum anderen sequentielle Informationen über die Textabschnitte, die annotiert wurden, enthält. Dieses erweiterte Datenmodell erlaubt eine parallele Sicht auf verschiedene Annotationsebenen.

Die annotierten Daten werden in eine Prolog-Faktenbasis übersetzt und es können Anfragen über die Struktur der Daten gestellt werden. Mittels verschiedener Prolog-Prädikate können Relationen sowohl zwischen Elementinstanzen als auch zwischen Elementklassen überprüft werden. Diese Informationen können zum einen dazu verwendet werden, Hypothesen über die Daten zu überprüfen und neue Information zu gewinnen, und zum anderen können die Informationen für eine Kombination der verschiedenen Informationsebenen verwendet werden.

In dem Projekt, in dessen Zusammenhang die vorgestellten Arbeiten stehen, hat neben dem Vergleich sprachlicher Funktionen mit sprachlichen Ausdrucksmitteln, auch zum Ziel Informationen, die auf verschiedene Ebenen verteilt sind, in eine gemeinsame Beschreibungsebene zu überführen. Eine Kombination verschiedener Beschreibungsebenen kann u.a. dann sinnvoll sein, wenn zu Beginn der Datenanalyse keine gemeinsame Dokumentgrammatik definiert werden konnte, da keine oder nur unvollständige Informationen darüber bekannt waren, wie die einzelnen Beschreibungsebenen zueinander in Beziehung stehen.

Ein weiterer Punkt für zukünftige Arbeiten ist die zusätzliche Berücksichtigung von Attributinformationen bei der Analyse von Beschreibungsebenen. Beispielsweise können für Elementklassen, denen nicht eindeutig eine Relation zugeordnet werden kann, und für die somit die Bildung einer gemeinsamen Beschreibungsebene nicht möglich ist, Unterklassen auf der Basis von Attributinformationen gebildet werden. Für diese Unterklassen besteht wiederum die Möglichkeit eindeutiger Relationen.

Literatur

Bonifati, Angela/Lee, Dongwong (2001): Technical Survey of XML Schema and Query Languages, (Submitted for journal publication).

Duruseau, Patrick/Brook O'Donnell, Matthew (2002): Concurrent Markup for XML Documents. XML Europe 2002, Barcelona.

Hartley, Anthony/Paris, Cécile (2001): Translation, controlled languages, generation. In: Steiner, Erich/Yallop, Colin (Hg.) Exploring Translation and Multilingual Text Production: Beyond Content. Berlin, New York: Mouton de Gruyter, 307-325.

Lehrndorfer, Anne (1996): Kontrolliertes Deutsch: linguistische und sprachpsychologische Leitlinien für eine (maschinell) kontrollierte Sprache in der technischen Dokumentation. Tübingen: Narr.

Renear, Allen/Mylonas, Elli/Durand, David (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.

Schachtl, Stephanie (1996): Requirements for Controlled German in Industrial Applications. In: Proceedings of the First International Workshop on Controlled Language Applications (CLAW 96), Leuven, 143-149.

Schröder, Bernhard (1998): Pro-SGML: Ein Prolog-basiertes System zum Textretrieval. In: Heyer, Gerhard/Wolff, Christian (Hg.): Linguistik und neue Medien. Wiesbaden: DUV, 205-216.

Simons, Gary F. (1998) The Nature of Linguistic Data and the Requirements of a Computing Envi-ronment for Linguistic Research. In: Lawler, John/Aristar-Dry, Helen (Hg.): Using Computers in Linguistics: A Practical Guide. London: Routledge. 10-25.

Sperberg-McQueen, C. M./Huitfeldt, Claus/Renear, Allen (2001): Meaning and interpretation of Markup. In: Markup Languages: Theory & Practice 2.3, MIT Press, 215-234.

Sperberg-McQueen, C. M./Huitfeldt, Claus/Dubin, David/Renear, Allen (2002): Drawing Inferences on the Basis of Markup. Proceedings of Extreme Markup Languages 2002, Montréal, Québec, August 2002.

Witt, Andreas (2002): Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie. Dissertation, Universität Bielefeld.