Strukturbezogene Klassifikation von Informationseinheiten in texttechnologischen Korpora

Felix Sasaki

1 Einleitung

Der vorliegende Artikel thematisiert Analyse- und Modellierungsmöglichkeiten für Informationseinheiten[1] in *texttechnologischen Korpora*. Vorgestellt wird eine Methode, Informationseinheiten anhand ihrer strukturell-positionellen Eigenschaften zu bestimmen. Im Zentrum steht die Frage, wie „strukturelle Eigenschaften“ zu definieren und zur Analyse einzusetzen sind. Der beschriebene Ansatz nutzt Pfadausdrücke zur *strukturellen Kontextbeschreibung* für ausgewählte Informationseinheiten, in einem deklarativen Format namens *context specification document* (CSD). Ein CSD-Dokument verwendet strukturelle Kontextbeschreibungen zur Klassifikation von Informationseinheiten.

Abschnitt 2 führt Eigenschaften texttechnologischer Korpora auf, die eine strukturbezogene, klassifizierende Sichtweise auf Informationseinheiten motivieren. In Abschnitt 3 werden grundlegende Begriffe von CSD definiert und Aufbau sowie Verarbeitung eines CSD-Dokuments beschrieben. Abschnitt 4 stellt ein konkretes Anwendungsszenario vor. In Abschnitt 5 werden Alternativen und Grenzen von CSD diskutiert. Es folgt eine Zusammenfassung und eine Reihe weiterführender Forschungsfragen.

2 Texttechnologische Korpora

Für texttechnologische Korpora lassen sich folgende, zentrale Eigenschaften aufführen:

1. Den Ausgangspunkt bilden Texte in elektronischer Form, z. B. Verschriftlichungen von Gesprächen.
2. Die Texte sind mit Informationen angereichert, d. h. auf verschiedenen Ebenen (z. B. linguistischen Beschreibungsebenen wie Morphologie, Syntax etc.) annotiert.
3. Die Informationsanreicherung nutzt Auszeichnungssprachen und Annotationskonventionen, oft formuliert in Schemasprachen wie XML-DTDs oder XML Schema (H. Thompson et al. 2001).

In (F. Sasaki und A. Witt (2003) wurden verschiedene Vorteile dargestellt, die sich aus diesen Eigenschaften für den linguistischen Bereich ergeben. Lin-

[1] Der Begriff ‚Informationseinheit’ wird in Abschnitt 3 genauer definiert.

guistische Ressourcen sind dank verbreiteter Auszeichnungssprachen und Annotationskonventionen leichter wiederverwendbar, große und diverse Datenmengen können erschlossen und miteinander verknüpft werden. Die Vielfalt unterschiedlicher Schriftsysteme wird durch standardisierte Zeichenkodierungssysteme in multilingualen Korpora repräsentierbar, und verschiedene Annotationsebenen können zueinander in Beziehung gesetzt werden.

Für ein texttechnologisches Korpus ergeben sich zudem Analyse- und Modellierungsmöglichkeiten, welche die *Dokumentstruktur* zum Gegenstand haben. Mit Standards wie XPath (J. Clark und S. deRose 1999), das der Beschreibung von Pfaden über Dokumentstrukturen dient, lassen sich *strukturbezogene Suchanfragen* formulieren wie „suche alle Nomen innerhalb einer Nominalphrase, die am Anfang eines Satzes steht, der der letzte Satz eines Paragraphen ist, welcher ...“. Auch speziell für linguistische Zwecke sind Abfragesprachen entwickelt worden, etwa im Rahmen des EU-Projektes MATE (A. Isard et al. 2000). Der vorliegende Artikel stellt das Format CSD vor, welches die strukturbezogene Verarbeitung von Informationseinheiten in texttechnologischen Korpora fokussiert. Unter „strukturbezogener Verarbeitung“ wird allerdings nicht eine Suchanfrage verstanden, sondern die Klassifikation der Informationseinheiten anhand strukturbezogener Kontextbeschreibungen.

3 Aufbau und Verarbeitung von CSD-Dokumenten

3.1 Grundlegende Begriffe

Im Folgenden wird das terminologische Basisinventar von CSD vorgestellt. *Informationseinheiten* sind die Einheiten, aus denen sich das XML Information Set zusammensetzt (J. Cowan und R. Tobin 2001). Sie bilden den Informationsgehalt eines XML-Dokuments[2]. Informationseinheiten sind z. B. das Dokument, ein Element, ein Attribut etc. Die Informationseinheit „Element“ wird u. a. benutzt, um Dokumentknoten zu benennen. Gegenstand der Kontextbeschreibung von CSD ist allein die Dokumentknotenstruktur, weshalb im Folgenden an Stelle von „Informationseinheit“ der Ausdruck „Element“ verwendet wird.

Ein CSD-Dokument klassifiziert nicht alle Elemente in einer XML-Dokumentinstanz[3], sondern nur eine bestimmte, anhand ihres Namens definierte Menge von Elementen (z. B. alle `p`-Elemente). Die Klassifikation besteht in der

[2] Das Information Set dient zwar primär der Beschreibung von Informationen in XML-Dokumenten. Die Informationseinheiten werden jedoch unabhängig von einer konkreten (XML)-Syntax definiert, so dass ihre Anwendung auch in anderen, in der Entwicklung befindlichen Auszeichnungssprachen möglich erscheint.

[3] Das Dokument, dessen Elemente mit einem CSD-Dokument klassifiziert werden, wird im Folgenden als Dokumentinstanz bezeichnet.

Bildung von Teilmengen, die ihrerseits unterteilt werden können. Die *Superklasse* umfasst diejenigen Elemente, die Gegenstand der Klassifikation sind, also z. B. „alle `p`-Elemente“. Der Ausdruck *Klasse* wird für die Benennung der Teilmengen benutzt[4]. Die Klassen sind der Superklasse in hierarchischer Form untergeordnet.

Die Klassifikation bzw. Teilmengenbildung wird mittels einer *Kontextbeschreibung* realisiert. Die Kontextbeschreibung erfolgt durch eine *strukturelle Bedingung*. Ein Elementknoten in der Dokumentinstanz gehört zu einer Klasse, wenn er die entsprechende strukturelle Bedingung erfüllt. Jede Klasse wird durch genau eine strukturelle Bedingung beschrieben. Die strukturelle Bedingung einer Klasse `p-0` kann z. B. lauten „Unmittelbar oberhalb vom `p`-Element muss das `div`-Element stehen“.

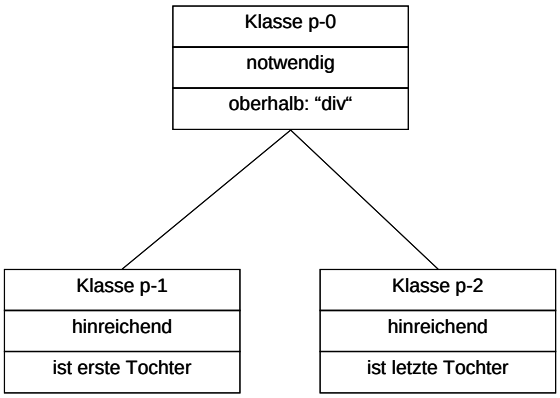

Abbildung 1: Strukturbezogene Klassifikation von Elementen

Die strukturelle Bedingung einer Klasse `p-1` lautet „Das `p`-Element muss das erste Tochterelement sein“, die Bedingung einer Klasse `p-2` „Das `p`-Element muss das letzte Tochterelement sein“. Strukturelle Bedingungen werden durch *Pfadausdrücke* formuliert, die Bewegungen und Tests über die Knotenstruktur der Dokumentinstanz sowie – optional – Tests von Elementnamen beinhalten.

Die Bedingungen werden unterschieden in *hinreichende* und *notwendige Bedingungen*. Dies ermöglicht es, bestimmte Klassen, also Teilmengen der Klassifikation, unter Beibehaltung der Klassifikationshierarchie „auszublen-

4 „Klasse“ wird hier *nicht* im Sinne des Paradigmas objektorientierter Programmierung (d. h. „Klasse“ als Vorlage für die Instanziierung der „Objekte“) verwendet.

den“. In Abbildung 1 wird die Bedingung der Klasse `p-0` als notwendig eingeordnet, die Bedingungen der Klassen `p-1` und `p-2` sind hinreichend[5].

Die strukturellen Bedingungen werden mittels Pfadausdrücken formuliert, die Elementnamenstests beinhalten, z. B. einen Test auf den Elementnamen „div“. Die Elementnamenstests greifen auf ein bestimmtes Inventar zurück, z. B. die Namen der HTML-Elemente. In einer separaten Definition werden Elementnamen, die einem anderen Inventar angehören, auf die im CSD-Dokument verwendeten Elementnamenstests eineindeutig abgebildet. Die Klassifikationen in einem CSD-Dokument sind auf diese Weise nicht an die Namenstests in den Pfadausdrücken gebunden. Eine Anwendung für die Abbildung von Elementnamen ist in Abschnitt 4.2 beschrieben.

3.2 Syntax eines CSD-Dokumenets

Der Aufbau eines CSD-Dokumentes[6] ist in Abbildung 2 wiedergegeben.

```
<csd mode="query">
<namespaceList>...</namespaceList>
 <superclass scope="p">
 <class id="p-0" sufficient="yes">
  <comment>Paragraph in einem Abschnitt</comment>
  <caterpillar>
   <up/>
   <element name="div"/>
  </caterpillar>
 </class>
 </superclass>
</csd>
```

Abbildung 2: Aufbau eines CSD-Dokumentes

CSD kann Namensräume berücksichtigen, die durch `namespace`-Elemente innerhalb des `namespaceList`-Elements definiert werden. Das `scope`-Attribut am `superclass`-Element enthält die Namen der zu klassifizierenden Elemente, in Abbildung 2 z. B. das `p`-Element. Ein `superclass`-Element enthält mindestens ein `class`-Element. Der Klassenname ist im `id`-Attribut des `class`-Elements angegeben. Das `sufficient`-Attribut mit dem Wert

5 Die Visualisierung stellt in vertikaler Abfolge für jede Klasse den Namen (z. B. `p-0`), die Einordnung als notwendig oder hinreichend sowie die strukturelle Bedingung (z. B. `oberhalb: "div"`) dar.

6 Beispieldokumente und Dokumentgrammatiken zur Erstellung von CSD-Dokumenten finden sich unter http://www.text-technology.de/csd/. Eine Implementierung in Python ist auf Anfrage erhältlich. Ein Onlineprozessor für CSD-Dokumente und Dokumentinstanzen findet sich unter http://kalk.lili.uni-bielefeld.de/caterpillar/csd.html .

`yes` spezifiziert die strukturelle Bedingung der Klasse als hinreichend. Steht kein `sufficient`-Attribut, gilt die Bedingung als notwendig. Die Klassifikationshierarchie wird durch Verschachtelung bzw. Nebenordnung der `class`-Elemente realisiert. Optional kann in die `class`-Elemente ein `comment`-Element zur informellen Beschreibung der Klasse eingefügt werden.

Die von CSD verwendete Pfadsprache zur Beschreibung der strukturellen Bedingungen sind so genannte Caterpillar-Ausdrücke[7], wie sie von (A. Brüggemann-Klein und D. Wood 2000) dargestellt werden. Ein Caterpillar-Ausdruck besteht aus bestimmten Bewegungen `up`, `left`, `right`, `first`, `last` über die Knotenstruktur einer Dokumentinstanz und Tests: `isFirst`, `isLast`, `isLeaf`, `isRoot`, `element`. Mit dem `element`-Element wird der Name des Knotens überprüft, auf dem sich der Caterpillar gerade befindet. Zusätzlich kann der Sternoperator, ausgedrückt durch das `zeroOrMore`-Element, angewendet werden.

3.3 Verarbeitung von CSD-Dokumenten und Dokumentinstanzen sowie mögliche Ausgaben

Das `mode`-Attribut des `csd`-Elements bestimmt den Verarbeitungsmodus. Im Modus `query` werden diejenigen Elemente einer Dokumentinstanz ausgegeben, die eine hinreichende Bedingungen erfüllen. Im Modus `validate` werden die Elemente ausgegeben, für die dies nicht der Fall ist. Zudem gibt es die Möglichkeit, in einer Dokumentinstanz an einzelnen Elementen ein `csd:caterpillar`-Attribut einzufügen[8], mit dem eine bestimmte Klassifikation für den Knoten vorgegeben wird. Erfüllt der Knoten in der Dokumentinstanz eine andere hinreichende Bedingung als die vorgegebene, so gilt er als nicht-valide.

Nach der Verarbeitung eines CSD-Dokuments und einer Dokumentinstanz kann zum einen ein Dokument wie in Abbildung 3 ausgegeben werden: Das Ausgabedokument enthält im `name`-Attribut des `document`-Elements den Namen der Dokumentinstanz, die verarbeitet wurde. Nach der optionalen Namensraumliste folgt für jede Superklasse, die im CSD-Dokument definiert ist, eine Liste von `nodelist`-Elementen. Sie führt in Form von `node`-Elementen diejenigen Knoten der Dokumentinstanz auf, die den im CSD-Dokument als Teil der Superklasse spezifizierten Elementnamen besitzen, z. B. alle `p`-Elemente. Für jeden Knoten wird im `path`-Element ein eindeutiger Pfad in XPath-Notation angegeben. Im Modus `query` folgt eine Liste der Klassen,

---

7 Die Vorteile der Caterpillar-Ausdrücke für CSD werden im Zuge eines Vergleichs mit anderen Ansätzen in Abschnitt 5 beschrieben.

8 Hierzu muss im Wurzelelement der Dokumentinstanz der csd-Namensraum `xmlns:csd="http://www.text-technology.de/csd"` definiert sein.

deren hinreichende Bedingungen durch den Knoten erfüllt werden. Im Modus `validate` steht stattdessen ein `error`-Element für diejenigen Knoten, die keine der hinreichenden Bedingungen erfüllen. Wenn ein Knoten zu anderen Klassen gehört als zu den im `csd:caterpillar`-Attribut angegebenen, führt ein `error`-Element die Klassen in einem `wrongClass`-Attribut auf.

```
<nodelists>
 <document name="corpus.xml"/>
  <namespaceList>...</namespaceList>
  <nodelist scope="p">
   <node path="/div/p[1]">
    <class id="is_first_child"/>
   </node>
  </nodelist>
</nodelists>
```

Abbildung 3: Eine mögliche Ausgabe nach der Verarbeitung eines CSD-Dokuments

Die beschriebenen Informationen können auch in die Dokumentinstanz integriert werden. Je nach Verarbeitungsmodus werden verschiedene Attribute eingefügt:

1. Im Modus `query` ein `csd:matches`-Attribut mit den Klassen, deren als hinreichend eingestufte Bedingungen vom jeweiligen Element erfüllt sind;
2. im Modus `validate` ein `csd:error`-Attribut mit dem Wert `“noMatch“` bei Elementen, die keine hinreichende Bedingung erfüllen, oder
3. im Modus `validate` ein `csd:wrongClass`-Attribut mit den Klassen, deren hinreichende Bedingungen erfüllt sind, die aber nicht im `csd:caterpillar`-Attribut stehen.

3.4 Verwendung der strukturellen Bedingungen für verschiedene Elementnamensinventare

CSD nutzt zur Beschreibung der strukturellen Bedingungen Elementnamenstests. Um die im CSD-Dokument formulierten strukturellen Bedingungen und Klassifikationen mit verschiedenen Inventaren von Elementnamen verbinden zu können, wird in einem separaten Dokument eine eineindeutige Abbildung des jeweiligen Namensinventars auf die Elementnamenstests formuliert (vgl. Abbildung 4). Das `group`-Element enthält mindestens ein `headElement`-Element, in dessen `name`-Attribut der Name eines Elementnamenstests aus dem CSD-Dokument steht. Es folgt mindestens ein `substitute`-Element. Darin steht ein `element`-Element, welches im `name`-Attribut einen Namen aus dem jeweiligen Inventar enthält. Bei der Verarbeitung dieser Liste werden die Ele-

mentnamen in den Dokumentinstanzen der jeweiligen Inventare entsprechend umbenannt, also z. B. das `m`-Element zu einem `syll`-Element oder das `pos`-Element zu einem `word-accent`-Element. Durch dieses Verfahren können Klassifikationen, die für Informationseinheiten einer Domäne entwickelt wurden – z. B. Prosodie –, auf andere Domänen – z. B. Morphosyntax – angewandt werden. Zugleich können domänen- oder sprachspezifische Klassifikationen generalisiert werden. Dabei werden spezifische Elementnamensinventare auf ein generelles Inventar von Elementnamenstests abgebildet (vgl. Abschnitt 4.2).

```
<substitutionList>
  <group id="morphosyntax">
    <headElement name="syll">
      <substitute id="morpheme">
        <element name="m"/>
      </substitute>
    </headElement>
    <headElement name="word-accent">
      <substitute id="word">
        <element name="pos"/>
      </substitute>
    </headElement>
  </group>
</substitutionList>
```

Abbildung 4: Abbildung eines Elementnamensinventars auf die Namenstests eines CSD-Dokuments

4 Anwendungsbeispiel: Sprachspezifische und sprachübergreifende Modellierung linguistischer Phänomene

Im Folgenden wird mit Hilfe von CSD dasselbe linguistische Phänomen in unterschiedlichen Sprachen exemplarisch modelliert[9]. Das Ziel besteht darin, das Phänomen *Kongruenz* zunächst in sprachspezifischen CSD-Dokumenten strukturbezogen zu klassifizieren. Anschließend werden Abbildungen sprachspezifischer Elementnamen auf sprachübergreifende Elementnamenstests definiert und mit diesen ein CSD-Dokument verfasst, das eine sprachübergreifende Klassifikation beinhaltet.

9 Weitere Anwendungsszenarien finden sich in (Sasaki und Pönninghaus 2003). Das hier vorgestellte Beispiel ist Teil einer Modellierung koreferentieller und anaphorischer Beziehungen im Japanischen. Sie entsteht im Rahmen des Projektes SEKIMO „Sekundäre Informationsstrukturierung und vergleichende Diskursanalyse“, ein Teilprojekt der Forschergruppe „Texttechnologische Informationsmodellierung“, vgl. http://www.text-technology.de .

4.1 Kongruenzphänomene im Deutschen und Japanischen

Kongruenz im weiteren Sinn kann definiert werden als die Übereinstimmung bestimmter kategorialer Eigenschaften sprachlicher Einheiten, insbesondere auf der Wortebene. Im engeren Sinne wird Kongruenz meist auf morphosyntaktische Phänomene bezogen, z. B. die Kongruenz von Subjekt und Verb im Deutschen wie in dem Satz „Ich heiße Sasaki“: Beiden Konstituenten lässt sich die Kategorie ‚erste Person Singular’ zuordnen. Die Problematik in der sprachübergreifenden Beschreibung liegt in der Tatsache begründet, dass in verschiedenen Sprachen unterschiedliche Kategorien von Kongruenzphänomenen betroffen sind. Zudem sind die Kategorien je nach Sprache auf anderen Beschreibungsebenen angesiedelt. Der folgende Satz ist eine mögliche japanische Übersetzung des obigen, deutschen Beispielsatzes, versehen mit einer Morphemglossierung:

```
(watashi ha)  Sasaki to   moushi   masu
ich     THEMA Sasaki PART heißen-  HONOR-ADD
minus-                    minus-   plus-
subhon                    subhon   addhon
```

Auch hier besteht eine Kongruenzbeziehung zwischen Subjekt `watashi` und Verb `moushimasu`, allerdings wird sie nicht – wie im Deutschen – durch morphosyntaktische Kategorien realisiert. Kongruenz zeigt sich im Höflichkeitssystem des Japanischen, welches auf verschiedenen Beschreibungsebenen Bestandteil der japanischen Gegenwartsprache ist. Auf der lexikalischen Ebene teilen das Subjekt `watashi` und das Verb `moushi` die Kategorie `minus-subhon`. Sie steht für eine ‚unhöfliche’ Einordnung des Subjekts. Aus Gründen, die auf einer pragmatisch-lexikalischen Ebene zu finden sind, muss ein Sprecher, wenn er selbst Subjekt des Satzes ist, diese Einordnung mittels der entsprechenden Lexeme für Subjekt und Verb vornehmen. Einfach gesagt gilt die Regel, sich selbst sprachlich ‚unhöflich’ zu behandeln. Auf einer morphologischen Ebene drückt das verbale Suffix `masu` die Kategorie `plus-addhon` aus, d. h. Höflichkeit gegenüber dem Hörer.

Anders als im Deutschen muss die Subjektposition im japanischen Beispiel nicht realisiert werden, weshalb das Subjekt im Beispielsatz geklammert steht. Es genügt das Verb mit der Kategorie `minus-subhon` und das verbale Suffix mit der Kategorie `plus-addhon`, um die Grammatikalität des Satzes zu gewährleisten. Auch ohne die konkreten Äußerungsbedingungen zu kennen ist es oft möglich, anhand der beschriebenen und anderer pragmatisch-lexikalischer Restriktionen die Subjektposition, d. h. den Antezedenten der sogenannten 0-Pronomina zu inferieren, ein Umstand, den zum Beispiel (M. Siegel 2000) in einer formalgrammatischen Analyse des Japanischen nutzt.

4.2 Einzelsprachliche und sprachübergreifende Modellierung der Kongruenz

Um die am Anfang von Abschnitt 4 formulierten Modellierungsziele erreichen zu können, werden zunächst partielle Annotationen der in Abschnitt 4.1 beschriebenen Ebenen vorgenommen:

1. morphologische Kategorie, z. B. `1Ps.Sg` für `e` von `heiße;`
2. Wortart, z. B. `Pronomen` für `ich`
3. syntaktische Funktion, z. B. `Prädikat` für `heiße;`
4. explizite Sprecherreferenz, z. B. `ich;`
5. nicht-realisierte Sprecherreferenz.

Der Unterschied zwischen (4) und (5) besteht in der Art, wie Sprecherreferenz mit sprachlich realisierten Einheiten verknüpft ist. (4) kann mit den sprachlichen Formen `ich` bzw. `watashi` verbunden werden, (5) ist nur eine Metainformation über den ganzen Satz. Abbildung 5 zeigt drei Dokumentinstanzen, in denen die fünf Ebenen annotiert sind: Eine deutsche Annotation, eine japanische mit expliziter Sprecherreferenz, und eine japanische mit impliziter Sprecherreferenz, d. h. ohne Realisierung des Subjekts.

Drei CSD-Dokumente, die die Kongruenzeigenschaften der `verb`-Elemente in diesen Dokumentinstanzen sprachspezifisch und sprachübergreifend klassifizieren, finden sich in Abbildung 6[10]. Das CSD-Dokument I klassifiziert die `verb`-Elemente der deutschen Annotation. In der Klasse `d-1` wird ein Pfadausdruck formuliert, der vom `verb`-Element zur verbalen, morphosyntaktischen Kategorie `m-1pers-sg` führt. In der Klasse `d-2` steht ein Pfadausdruck, der über das `subjekt`-Element zur Subjektkategorie `m-1pers-sg` führt, in der Klasse `d-3` ein Pfadausdruck, der bei gleichem Ziel spezifischer ist als der Pfadausdruck aus Klasse `d-2`. Das CSD-Dokument II klassifiziert die `verb`-Elemente in den japanischen Beispielen. Die Klasse `j-1` enthält einen Pfad zum `minus-subjekt-hon`-Element, also zur beschriebenen pragmatisch-lexikalischen Restriktion. Die strukturelle Bedingung in der Klasse `j-2` ist durch beide japanischen Beispiele erfüllbar. Die Bedingungen in den Klassen `j-3` und `j-4` hingegen werden jeweils durch die explizite (Klasse `j-4`) oder die implizite Referenz (Klasse `j-3`) erfüllt. Das CSD-Dokument III enthält eine Klassifikation der `verb`-Elemente, die Kongruenz sprachübergreifend modelliert. Diese Modellierung ist durch die Generalisierung des Elementnameninventars realisiert (vgl. Abschnitt 3.4). Die `minus-subjekt-hon`-Elemente

10 Die Unterscheidung in hinreichende und notwendige Bedingungen ist für die vorliegende Modellierung irrelevant und wird in Abbildung 6 nicht visualisiert. Die Elementnamenstests sind durch Anführungsstriche gekennzeichnet.

und `m-1pers-sg`-Elemente in den Dokumentinstanzen werden auf `kongruenz-marker`-Elemente abgebildet. Die strukturellen Bedingungen in den Klassen `jd-1` und `jd-2` werden durch die explizite Referenz im Japanischen und im Deutschen erfüllt, die strukturelle Bedingung in der Klasse `j-3` ist aus dem CSD-Dokument II übernommen und wird nur vom Japanischen erfüllt.

```
<corpus>
<s>
 <sprecherreferenz>
  <subjekt><pronomen>
   <m-1pers-sg>Ich</m-1pers-sg>
  </pronomen></subjekt>
 </sprecherreferenz>
 <praedikat><verb>
  <verb-stem>heiß<m-1pers-sg>e</m-1pers-sg>
  </verb-stem>
 </verb></praedikat>Sasaki
</s>
<s>
 <sprecherreferenz>
  <subjekt><pronomen>
   <minus-subjekt-hon>watashi</minus-subjekt-hon>
  </pronomen></subjekt>
 </sprecherreferenz> ha Sasaki to
 <praedikat><verb>
  <verb-stem><minus-subjekt-hon>moushi
  </minus-subjekt-hon>masu</verb-stem>
 </verb></praedikat>
</s>
<s>
 <implizite-sprecherreferenz>Sasaki to
  <praedikat><verb>
     <verb-stem><minus-subjekt-hon>moushi
     </minus-subjekt-hon>masu</verb-stem>
  </verb></praedikat>
 </implizite-sprecherreferenz>
</s>
</corpus>
```

Abbildung 5: Deutsche und japanische Annotation von Kongruenzphänomenen

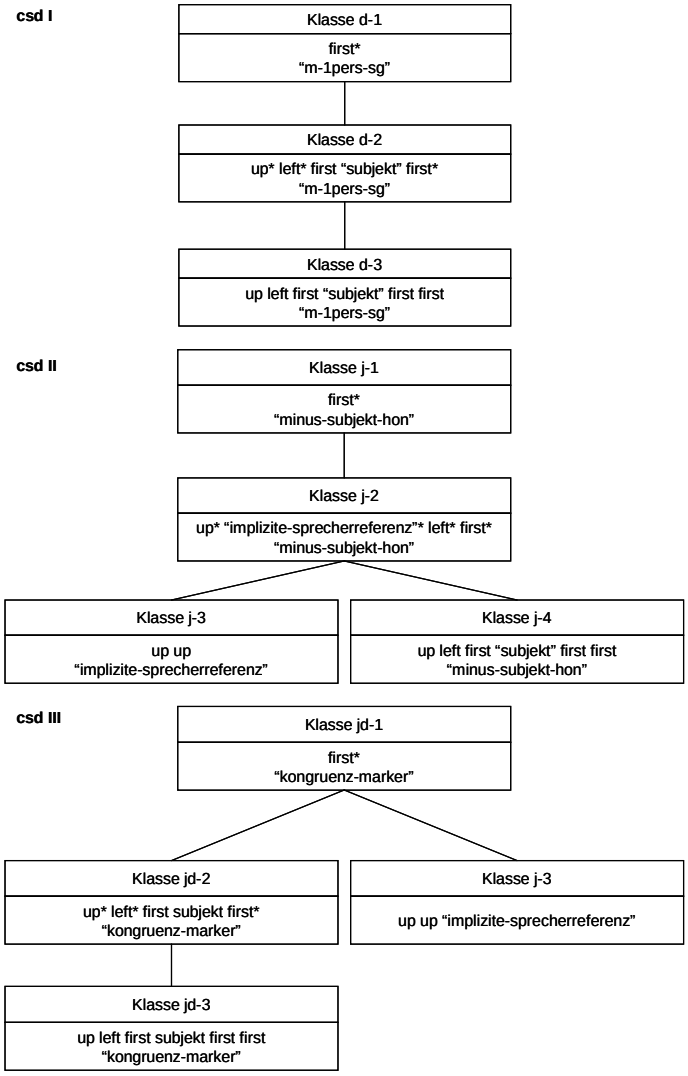

Abbildung 6: Sprachspezifische und sprachübergreifende Klassifikationen

Die dargestellte Klassifikation ist anhand sehr einfacher Beispiele entwickelt worden. In einem weiteren Schritt kann sie an größeren, texttechnologischen Korpora getestet und verfeinert werden. Für die behandelte Domäne scheint es nützlich zu sein, die Mächtigkeit der Caterpillarausdrücke um einen Optionali-

tätsoperator zu erweitern, z. B. für die Modellierung der dokumentstrukturellen Distanz kongruierender Einheiten.

5 Alternativen und Grenzen des Ansatzes

Die Funktionalität von CSD, in einem Format zugleich Strukturen von Dokumentinstanzen zu analysieren und zu restringieren, verbindet Eigenschaften von Query- und Schemasprachen. Schemasprachen fokussieren eine hierarchische Strukturdeklaration, die für eine Klasse von Dokumenten gilt. CSD stellt diesem Ansatz eine knoten- und pfadzentristische, auf Dokumentinstanzen ausgerichtete Perspektive gegenüber.

Die Schemasprache XML Schema (H. Thompson et al. 2001) erlaubt durch die Verwendung der `selector`- und `field`-Elemente die Formulierung nicht-hierarchischer Restriktionen für bestimmte Informationseinheiten. Zunächst wird im `selector`-Element durch einen XPath-Ausdruck eine Menge von Informationseinheiten (Elemente, Attribute, ...) ausgewählt. Die Bedingung, welche diese Informationseinheiten erfüllen müssen, wird in einem oder mehreren `field`-Elementen wieder mit einem XPath-Ausdruck beschrieben. XML Schema verwendet also wie CSD ein gestuftes Verfahren: (1) Auswahl einer Menge von Informationseinheiten; (2) Beschreibung einer Bedingung, welche diese Informationseinheiten erfüllen müssen. Diese zweistufige Vorgehensweise kennzeichnet auch Schematron (vgl. R. Jeliffe). Zunächst wird mittels eines XPath-Ausdrucks eine Menge von Informationseinheiten spezifiziert, anschließend wird ein Test für diese Einheiten – ebenfalls mittels XPath – formuliert. Die Möglichkeiten von Schematron und den beschriebenen Konstruktionen in XML Schema gehen insofern über CSD hinaus, als (eine Teilmenge von) XPath für beide Stufen genutzt wird. D. h., nicht nur Elementnamen, sondern auch andere Informationseinheiten (Attributnamen, -werte, ...) sind potentieller Bestandteil der Pfadausdrücke. CSD verwendet nur die Informationseinheit ‚Element', da so die beschriebene Abbildung der Namenstests und die flexible Anwendung der strukturellen Bedingung gewährleistet ist. Dies ist auch der Grund, warum CSD Caterpillar-Ausdrücke einsetzt. Zudem gibt es weder in XML Schema noch in Schematron die Möglichkeit, die Erfüllbarkeit von Bedingungen als Klassifikationskriterium für Informationseinheiten anzuwenden.

Genau diese Option bietet XQuery (vgl. S. Boag et al. 2002), eine Querysprache, die sich im Prozess der Verabschiedung durch das World Wide Web Consortium befindet. XQuery erlaubt es, in Suchanfragen Informationen aus einem XML Schema Dokument zu verwenden. In einem XML-Schema-Dokument können Typenhierarchien definiert werden. Es ist z. B. möglich, einen Elementtyp `adresse-allgemein` (enthält `name, strasse` und `ort`) zu definieren und andere Typen, z. B. `deutsch` (enthält zusätzlich eine `post-leitzahl`) und amerikanisch (enthält zusätzlich einen `zip-code`) davon

abzuleiten. Diese Typenhierarchie ähnelt der hierarchischen Klassifikation in CSD-Dokumenten. In XML-Schema und XQuery wird im Gegensatz zu CSD jedoch eine starke Typisierung verwendet, d. h. in der Typenhierarchie darf es keine Mehrdeutigen bei der Typzugehörigkeit geben. Das Vorgehen von CSD hingegen kann als schwache Typisierung bezeichnet werden: Eine Informationseinheit kann mehreren Klassen angehören.

Der Ansatz, Restriktionen über bestimmten Informationseinheiten in einem zur eigentlichen Dokumentinstanz separaten Dokument zu formulieren, wird auch von der *Text Encoding Initiative* (TEI; C. M.Sperberg-McQueen und L. Burnard 1994) in der *Feature System Declaration* (FSD, Kapitel 26) verfolgt. Die FSD deklariert Restriktionen über Dokumenten, in denen Merkmalsstrukturen (Kapitel 16 der TEI) annotiert sind. Es können Restriktionen in Bezug auf die Wertebereiche von Daten, Merkmalstypen bestimmter Merkmalsstrukturen, oder die Kookkurrenz von Merkmals-Wert Paaren formuliert werden. Die FSD macht anders als CSD, Schematron oder XML Schema keinen Gebrauch von bestimmten Pfadsprachen. Eine interessante, noch zu klärende Frage lautet deshalb, in welchem Verhältnis die Restriktionen der FSD zu denen von CSD stehen.

Der CSD-Ansatz setzt Dokumentinstanzen voraus, in denen (linguistische) Strukturen hierarchisch repräsentiert sind. Zur CSD-basierten Modellierung muss also bekannt sein, welche Verhältnisse zwischen den verschiedenen (linguistischen) Beschreibungsebenen in der Dokumentinstanz bestehen. Dies zeigt sich z. B. an den Pfadausdrücken aus Abschnitt 4.2, die eine bestimmte Hierarchisierung der Kategorien voraussetzen: Referenz > syntaktische Funktion > Wortart > morphologische Kategorie. Diese Hierarchisierung basiert jedoch nicht auf einer „unumstößlichen“ linguistisch-theoretischen Rechtfertigung. Für den CSD-Ansatz erscheint es deshalb als sinnvoll, auf einer expliziten Repräsentation von Beziehungen zwischen (linguistischen) Beschreibungsebenen aufzubauen. Eine Diskussion geeigneter Methoden findet sich in (A. Witt in diesem Band).

6 Zusammenfassung und Ausblick

Der vorliegende Artikel stellte einen Ansatz zur Beschreibung strukturbezogener Eigenschaften von Informationseinheiten in Dokumentinstanzen vor. CSD ermöglicht es, Informationseinheiten in einer Dokumentinstanz anhand der Erfüllbarkeit struktureller Bedingungen, die mittels Pfadausdrücken formuliert werden, zu klassifizieren. Die Option, beliebige Elementnamensinventare auf die Namenstests in den Pfadausdrücken abzubilden, ermöglicht die Verbindung einer Klassifikation mit verschiedenen dokumentgrammatischen Namensinventaren. Dadurch kann die Klassifikation auch generalisiert werden. Im vorgestellten Beispiel wurde die Methodik auf ein linguistisches Phänomen angewandt,

für das die Formulierung und Generalisierung strukturbezogener Klassifikationen über verschiedene Domänen oder Sprachen hinweg sehr vielversprechend erscheint.

Um die Abbildbarkeit der Elementnamensinventare in einfacher Form zu gewährleisten, wurde die Pfadsprache der Caterpillar-Ausdrücke bei der Erstellung von CSD-Dokumenten genutzt. Für die Zukunft bleibt zu testen, welche Veränderungen diese Sprache erlaubt, ohne die Funktionalität von CSD zu beeinträchtigen. Für das dargestellte, linguistische Phänomen könnte insbesondere die Verwendung optionaler Ausdrücke nützlich sein. Da CSD als Komplement zu Schemasprachen angesehen werden kann, liegt eine weitere Forschungsaufgabe in der Klärung der formalen Beziehungen zwischen Schemasprachen und Caterpillarausdrücken, wie sie von CSD verwendet werden. Sinnvoll erscheint auch die Einbindung weiterer Informationseinheiten in die Pfadausdrücke, z. B. Attributnamen und -werte sowie bestimmte Attribut-Wert Kombinationen. Allerdings muss bei der Erweiterung von CSD darauf geachtet werden, dass die Beziehung zu Schemasprachen und die Abbildbarkeit von Informationseinheitstests nicht verloren gehen.

7 Literatur

Boak, Scott / Chamberlin, Don / Fernandez, F. Mary / Florescu, Daniela / Robie, Jonathan / Jérôme, Siméon (2002): XQuery 1.0: An XML Query Language. W3C-Working Draft, 15. November 2002. Siehe http://www.w3.org/TR/xquery/

Brüggemann-Klein, Anne / Wood, Derick: Caterpillars: A Context Specification Technique. In: Markup Languages: Theory & Practice 2(1). 81-106

Clark, James / deRose, Steve (1999): XML Path Language (XPath) Version 1.0. W3C-Empfehlung, 16. November 1999. Siehe http://www.w3.org/TR/XPath/

1Cowan, John / Tobin, Richard (2001): XML Information Set. W3C-Empfehlung, 24. November 2001. Siehe http://www.w3.org/TR/xml-infoset/

Isard, Amy / McKelvie, David / Mengel, Andreas / Møller, Morton B. / Grosse, Michael / Olsen, Martin V. (2000): MATE Deliverable 3.2: Data Structures and APIs for the MATE Workbench. Kapitel 3: Query Language. Siehe http://www.ltg.ed.ac.uk/~amyi/mate/D3.2/

Jeliffe, Rick: Schematron. An XML Structure Validation Language using Patterns in Trees. Siehe http://www.ascc.net/xml/resource/schematron/schematron.html

Lobin, Henning / Lemnitzer, Lothar (Hrsg.) (2003): Texttechnologie. Perspektiven und Anwendungen. Tübingen: Stauffenburg-Verlag

Sasaki, Felix (2003): Textauszeichnung im Original und in der Übersetzung: Schemasprachen und mehr. In: Proceedings der GLDV-Frühjahrstagung 2003

Sasaki, Felix / Pönninghaus, Jens (2003): Testing structural properties in textual data: beyond document grammars. In: Literary and Linguistic Computing. Special issue dedicated to the ALLC/ACH conference 2002

Sasaki, Felix / Witt, Andreas (2003): Linguistische Korpora. In: Lobin und Lemnitzer (2003)

Siegel, Melanie (2000): HPSG Analysis of Japanese. In: Wahlster, W. (2000)

Sperberg-McQueen, Michael C. / Burnard, Lou (1994): Guidelines for Electronic Text Encoding and Interchange (TEI P3). Chicago and Oxford: Text Encoding Initiative

Thompson, Henry / Beech, David / Murray, Maloney / Mendelsohn, Noah (2001): XML Schema Part 1: Structures. W3C-Empfehlung, 2 Mai 2001. Siehe http://www.w3.org/TR/xmlschema-1/

Wahlster, Wolfgang (Hrsg.) (2000): Verbmobil: Foundations of Speech-to-Speech Translation. Berlin: Springer

Witt, Andreas (in diesem Band): Linguistische Informationsmodellierung