Eine XML-Kodierung für AVM-Beschreibungen

Manfred Sailer und Frank Richter*

Seminar für Sprachwissenschaft, Wilhelmstr. 113, 72074 Tübingen
{sailer,fr}@sfs.nphil.uni-tuebingen.de

Zusammenfassung Ausgehend von der Beobachtung, dass existierende Implementierungsplattformen für HPSG-Grammatiken untereinander inkompatible interne Datenformate für Attribut-Wert-Matrizen (AVMs) wählen, schlägt das vorliegende Papier eine XML-basierte Beschreibung von AVMs vor. Diese Normierung von AVMs ist geeignet, als Transferformat an den Schnittstellen existierender HPSG-Systeme zu fungieren und somit die Problematik miteinander unvereinbarer Datenstrukturen zu lindern.

1 Einleitung

Im Bereich der HPSG (Head-Driven Phrase Structure Grammar, [13]) ist bereits eine Reihe von Datenbanken vorhanden oder im Aufbau [3, 6, 7, 8, 10], und es gibt Implementierungsplattformen mit Grammatiken mit relativ hoher empirischer und/oder theoretischer Abdeckung wie ALE [2], Babel, LKB [9], PAGE, ConTroll, Trale [12]. Aufgrund der unterschiedlichen internen Datenformate, die diese Systeme verwenden, wäre es wünschenswert, ein einheitliches und der maschinellen Weiterverarbeitung leicht zugängliches Format zu definieren, in das die system-abhängigen Kodierungen übersetzt werden können. Hierfür bietet sich die *Extensible Markup Language* (XML) an.

Für linguistische Datenbanken gibt es bereits Bemühungen, ein XML-Format zu finden, das den Austausch von Daten erleichtert. So wird in [11] gezeigt, wie verschiedene Arten von Baumrepräsentationen in XML enkodiert werden können. Daneben gibt es in [1] einen Vorschlag zur XML-Enkodierung von Strukturen der Tree-Adjoining Grammar (TAG). Dabei handelt es sich um Bäume, die Attribut-Wert-Paare als Knotenlabels haben können; außerdem werden einem Satz zum Teil mehrere Bäume simultan zugeordnet.

Für die HPSG ist keine dieser Kodierungsformen geeignet. Die Ausgabe von HPSG-Parsern sind in der Regel komplexe Attribut-Wert-Matrizen (*attribute-value matrice, AVM*). AVMs sind im HPSG-Verständis *Beschreibungen* von linguistischen Objekten (vergl. [14]). Im Gegensatz dazu sind Bäume, auch solche mit komplexen Knotenlabels wie in TAG, *Modelle* von linguistischen Objekten.

* Wir danken Maria Atanasowa, Lothar Lemnitzer, Zdravko Peev, Kiril Simov, Ilona Steiner und Tylman Ule für Kommentare und Diskussion. Eine ausführlichere Version des vorliegenden Papiers ist elektronisch erhältlich unter der URL www.sfs.nphil.uni-tuebingen.de/~mf/papers/index.html.

Damit haben Baumbanken prinzipiell einen anderen ontologischen Status als “AVM”-Banken.[1]

Die verwendeten AVMs sind rekursive *Klammerstrukturen*, in denen neben atomaren Werten auch AVMs oder Tags auftreten können. Abbildung 1 zeigt eine einfache AVM, wobei Attributnamen in Kapitälchen erscheinen (SYNSEM), Sortennamen kursiv geschrieben sind (*phrase*) und Tags als Zahlen in Boxen dargestellt werden ($\boxed{1}$). Diese AVM beschreibt all diejenigen Objekte, die das Label *phrase* tragen, und für die gilt, dass die Kante SYNSEM zu einem Objekt führt, von dem aus eine Kante HEAD zu einem weiteren Objekt mit dem Label *verb* führt. Dieses letztgenannte Objekt mit Label *verb* wird ebenfalls erreicht über die Kanten HEAD-DTR, SYNSEM und HEAD. Durch den Tag $\boxed{1}$ wird diese Identität ausgedrückt.

$$\begin{bmatrix} \textit{phrase} & \\ \textsc{synsem} & \begin{bmatrix}\textsc{head} & \boxed{1}\ \textit{verb}\end{bmatrix} \\ \textsc{head-dtr} & \begin{bmatrix}\textsc{synsem} & \begin{bmatrix}\textsc{head} & \boxed{1}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Abbildung 1. Eine einfache AVM

Dass es sich bei einer AVM um eine Beschreibung und nicht um ein linguistisches Objekt selbst handelt, sieht man beispielsweise daran, dass in Abbildung 1 keine Phonologie angegeben ist, obwohl jedes Objekt der Sorte *phrase* einen PHONOLOGY-Wert hat.

AVMs sind nur ein Teil der HPSG-Beschreibungssprache: Auf HPSG-Plattformen wie ConTroll und Trale können neben den eigentlichen AVM-Klammerstrukturen auch andere Elemente der Beschreibungssprache im Input und im Output erscheinen. [14] definiert eine AVM-Beschreibungssprache für HPSG, die auch *relationale Ausdrücke*, *logische Junktoren* und *Quantoren* umfasst. In Abbildung 2 ist eine solche Beschreibung gegeben, die aus einer Klammerstruktur, dem Junktor “$\wedge$” (Konjunktion), sowie einer Relation (`append`) besteht.[2] Das nicht-logische Vokabular von Beschreibungen (Attribut-, Sorten- und Relationsnamen, sowie die Stelligkeit von Relationen) wird wie in logischen Sprachen wie zum Beispiel in Sprachen der Prädikatenlogik erster Stufe üblich über eine *Signatur* festgelegt. Das logische Vokabular hingegen (Tags und Junktoren) ist signatur-unabhängig.[3]

[1] [5] argumentiert, dass es für HPSG-basiertes Parsen sinnvoll ist, nicht Modelle zu konstruieren, sondern Beschreibungen zu finden.

[2] Wir ignorieren Quantoren und interpretieren im Folgenden alle Variablen (Tags) als existentiell abgebunden.

[3] Darüberhinaus enthält eine HPSG-Signatur die Sortenhierarchie und die Attributsdeklarationen. Diese Informationen sind wichtig zur Festlegung der Ontologie linguistischer Objekte, nicht jedoch für die Definition der Beschreibungssprache.

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{PHON} & \boxed{2} \\
\text{SYNSEM} & \begin{bmatrix}\text{HEAD} & \boxed{1}\ \textit{verb}\end{bmatrix} \\
\text{NONHEAD-DTR} & \begin{bmatrix} \text{PHON} & \boxed{3} \begin{bmatrix}\textit{nelist} & \\ \text{HD} & \textit{John} \\ \text{TL} & \textit{elist}\end{bmatrix} \\ \text{SYNSEM} & \begin{bmatrix}\text{HEAD} & \textit{noun}\end{bmatrix}\end{bmatrix} \\
\text{HEAD-DTR} & \begin{bmatrix} \text{PHON} & \boxed{4} \begin{bmatrix}\textit{nelist} & \\ \text{HD} & \textit{walks} \\ \text{TL} & \textit{elist}\end{bmatrix} \\ \text{SYNSEM} & \begin{bmatrix}\text{HEAD} & \boxed{1}\end{bmatrix}\end{bmatrix}
\end{bmatrix}
\quad \wedge \quad \texttt{append}(\boxed{3},\boxed{4},\boxed{2})
$$

Abbildung 2. Eine AVM mit Junktor und relationalem Ausdruck

In diesem Papier stellen wir die in Abbildung 3 gezeigte allgemeine Architektur für die XML-Kodierung von AVM-Beschreibungen vor. Da alle Signaturen aufgrund ihrer Definition dieselbe Gestalt haben, geben wir eine DTD für Signaturen vor, die in der Datei `signature.dtd` angegeben ist. Die konkrete Signatur einer Grammatik, das heißt eine Signatur mit bestimmten Attributen, Sorten und Relationen, ist dann als eine Instanz dieser DTD zu verstehen und als XML-Datei `signature.xml` kodiert. Da die Signatur die Beschreibungssprache genau festlegt, können wir aus der Datei `signature.xml` eine DTD für AVM-Beschreibungen, `avm.dtd`, erzeugen. Hierzu verwenden wir XSL-Transformationen, `sig2avm.xsl`. Instanzen der DTD `avm.dtd`, in der Abbildung als `avm.xml` angegeben, sind XML-Kodierungen von AVM-Beschreibungen. Diese Beschreibungen können in ihrer linguistischen Anwendung verschiedenerlei Funktionen erfüllen. Sie können Prinzipien der Theorie einer Grammatik sein; oder relativ allgemeine Beschreibungen linguistischer Objekte, so wie sie als Anfragen an HPSG-Parser gegeben werden; oder relativ exakte Beschreibungen linguistischer Objekte, wie sie als Output von HPSG-Parsern geliefert werden.

$$
\begin{tikzcd}
\texttt{signature.dtd} \arrow[d, "Instanz"'] & & \\
\texttt{signature.xml} \arrow[r, "\texttt{sig2avm.xsl}"] & \texttt{avm.dtd} \arrow[d, "Instanz"'] \\
 & \texttt{avm.xml}
\end{tikzcd}
$$

Abbildung 3. Überblick

Im vorliegenden Papier gehen wir nur auf die Dateien `avm.dtd` und `avm.xml` ein, und nehmen eine feste Signatur an. In der ausführlicheren Version des Papiers werden auch die Signaturdateien erläutert.

2 Die Kodierung von AVM-Beschreibungen

In Abbildung 4 ist eine automatisch erzeugte DTD für AVM-Beschreibungen gegeben. Diese DTD besteht aus einem signatur-spezifischen Teil und einem signatur-unabhängigen Teil. Im ersten Teil werden die in der Signatur eingeführten Namen der Sorten, Attribute und Relationen als mögliche Werte der parametrisierten Entitäten `%sorts;`, `%attrs;` und `%relation;` aufgeführt. Für jede Relation, hier nur `append`, wird außerdem ein Element eingeführt, das die Stelligkeit der Relation ausdrückt.

Wir definieren ein Element `descr`, das aus einer beliebig langen Folge von Klammerstrukturen, relationalen Ausdrücken oder logischen Verknüpfungen besteht. Klammerstrukturen werden durch `avm`-Elemente kodiert, relationale Ausdrücke durch Elemente der parametrisierten Entität `%relation;` und Junktoren durch Elemente der Entität `%constant;`. Abbildung 5 zeigt die XML-Kodierung der AVM von Abbildung 1.

In unserer DTD haben wir die an sich rekursive Gestalt von AVMs aufgebrochen. Dadurch wird die XML-Kodierung flach und spiegelt die Einbettungen von Klammerstrukturen nicht direkt wider. Andererseits können Tags auf einfache Weise als Identifizierer (Werte vom Typ `IDREF`) kodiert werden. Da Tags in AVMs extrem häufig sind, ist diese Kodierungsform wohl motiviert. Außerdem ermöglicht die flache Kodierung mehr Freiheiten hinsichtlich der graphischen Darstellung von AVMs: die XML-Kodierung in Abbildung 5 kann graphisch dargestellt werden wie in Abbildung 1 oder wie in Abbildung 6

Abbildung 7 zeigt die XML-Kodierung der umfangreicheren AVM-Beschreibung von Abbildung 2. Man beachte den relationalen Ausdruck in der vorletzten Zeile und das Junktorelement `and`.

3 Interaktion mit HPSG-Tools

Die Nützlichkeit der XML-Kodierung ergibt sich aus der Anbindung an bereits existierende HPSG-Werkzeuge. Zur graphischen Darstellung von AVM-Beschreibungen kann aus dem XML-Format automatisch in das Datenformat von Grisu (GRammar vISUalization tool, [15]) konvertiert werden. Grisu erlaubt viele benutzerdefinierte Eingriffsmöglichkeiten in die Darstellungsform einer Klammerstruktur, wie sie bei der Grammatikentwicklung benötigt werden.

Bislang wird Grisu hauptsächlich als Ausgabetool für die HPSG-Implementierungsplattform Trale [4, 12] verwendet. Um die Ausgabedaten von Trale auch in XML verfügbar zu machen, soll eine Konversion aus dem Grisu-Format in das XML-Format implementiert werden.

```
<!--     automatisch erzeugte DTD fuer AVM-Beschreibungen     -->

<!--    ####################################################   -->
<!--              Signatur-spezifischer Teil                   -->

<!--                     Sorten                                -->
<!ENTITY % sorts        "(sign|word|phrase|synsem|
                          list|e_list|ne_list)"                  >


<!--                     Attribute                             -->
<!ENTITY % attrs        "(phon|synsem|h_dtr|n_dtr|hd|tl)"        >

<!--                     Relationen                            -->
<!ENTITY % relation     "(append)"                               >

<!--                     einzelne Relationen                   -->
<!ELEMENT  append       (arg,arg,arg)                            >

<!--    ####################################################   -->
<!--              Signatur-unabhaengiger Teil                  -->

<!--                     Beschreibung                          -->
<!ELEMENT  descr        (avm|%relation;|%constants;)+            >

<!--                     AVM-Klammerstruktur                   -->
<!ELEMENT  avm          (avp*)                                   >
<!ATTLIST  avm
            top         ID                            #REQUIRED
            sort        %sorts;                       #IMPLIED   >

<!                       Attribut-Wert-Paar                    -->
<!ELEMENT  avp          EMPTY                                    >
<!ATTLIST  avp
             attr       %attrs;                       #REQUIRED
             value      IDREF                         #REQUIRED  >

<!--                     Argument (fuer relationale Beschr.)   -->
<!ELEMENT  arg          EMPTY                                    >
<!ATTLIST  arg
             top        IDREF                         #REQUIRED  >

<!--                     logische Konstanten                   -->
<!ENTITY % constant     "(and|or|if|not)"                        >

<!--                     Konjunktion                           -->
<!ELEMENT and           (descr,descr)                            >

<!--                     Disjunktion                           -->
<!ELEMENT or            (descr,descr)                            >

<!--                     Implikation                           -->
<!ELEMENT if            (descr,descr)                            >

<!--                     Negation                              -->
<!ELEMENT not           (descr)                                  >
```

Abbildung 4. Eine signatur-spezifische DTD für AVM-Beschreibungen

```
<descr>
<avm sort="phrase">                                <!-- Phrase -->
     <avp attr="synsem" value="id02"/>
     <avp attr="head-dtr" value="id03"/>
</avm>

<avm top="id02">
     <avp attr="head" value="id01"/>
</avm>

<avm top="id01" sort="verb"/>

<avm top="id03">                              <!-- Kopftochter -->
     <avp attr="synsem" value="id04"/>
</avm>

<avm top="id04">
     <avp attr="head" value="id01"/>
</avm>
</descr>
```

Abbildung 5. XML-Kodierung der AVM in Abbildung 1

$$\text{Alternative 1: } \begin{bmatrix} \textit{phrase} & \\ \textsc{synsem} & \begin{bmatrix}\textsc{head} & \boxed{1}\end{bmatrix} \\ \textsc{head-dtr} & \begin{bmatrix}\textsc{synsem} & \begin{bmatrix}\textsc{head} & \boxed{1}\ \textit{verb}\end{bmatrix}\end{bmatrix} \end{bmatrix}$$

$$\text{Alternative 2: } \begin{bmatrix} \textit{phrase} & \\ \textsc{synsem} & \begin{bmatrix}\textsc{head} & \boxed{1}\ \textit{verb}\end{bmatrix} \\ \textsc{head-dtr} & \begin{bmatrix}\textsc{synsem} & \begin{bmatrix}\textsc{head} & \boxed{1}\ \textit{verb}\end{bmatrix}\end{bmatrix} \end{bmatrix}$$

Abbildung 6. Zwei Darstellungsalternativen zu Abbildung 1

```
<descr>
  <and>
    <descr>                                         <!-- Klammerstruktur -->
      <avm sort="phrase">                               <!-- Gesamtphrase -->
           <avp attr="phon" value="id02"/>
           <avp attr="synsem" value="id05"/>
           <avp attr="nonhead-dtr" value="id06"/>
           <avp attr="head-dtr" value="id07"/>
      </avm>

      <avm top="id05"><avp attr="head" value="id01"/></avm>
      <avm top="id01" sort="verb"/>

      <avm top="id06">                             <!-- Nichtkopftochter -->
           <avp attr="phon" value="id03"/>
           <avp attr="synsem" value="id08"/>
      </avm>
      <avm top="id03" sort="nelist">
           <avp attr="hd" value="id09"/>
           <avp attr="tl" value="id10"/>
      </avm>
             <avm top="id09 sort="John"/>
      <avm top="id10 sort="elist"/>
      <avm top="id08">
           <avp attr="head" value="id11"/>
      </avm>
      <avm top="id11 sort="noun"/>

      <avm top="id07">                                     <!- Kopftochter ->
           <avp attr="phon" value="id04"/>
           <avp attr="synsem" value="id12"/>
      </avm>
      <avm top="id04" sort="nelist">
           <avp attr="hd" value="id13"/>
           <avp attr="tl" value="id14"/>
      </avm>
      <avm top="id13 sort="walks"/>
      <avm top="id14 sort="elist"/>
      <avm top="id12">
           <avp attr="head" value="id01"/>
      </avm>
    </descr>

    <descr>                                 <!-- relationaler Ausdruck -->
      <append><arg top="id03"/><arg top="id04"/><arg top="id02"/></append>
    </descr>
  </and>
</descr>
```

Abbildung 7. XML-Kodierung von Abbildung 2

Allgemeiner gesehen eignet sich das XML-Format als Datenaustauschformat zwischen verschiedenen Systemen, die AVMs intern in unterschiedlicher Weise repräsentieren. Eine denkbare Anwendung der XML-Kodierung von AVM-Beschreibungen wäre beispielsweise ihre Verwendung beim Erstellen von Grammatik-Testsuiten, die systemunabhängig verwendet werden sollen. Die Testsequenzen können im XML-Format kodiert und dann in das Input-Format des jeweiligen Parsers, zum Beispiel Trale, konvertiert werden. Umgekehrt kann auch aus dem Outputformat von Trale nach XML konvertiert werden. Die hier vorgestellte Kodierung von AVM-Beschreibungen eignet sich damit hervorragend für HPSG-orientiertes NLP.

4 Zusammenfassung

In diesem Papier haben wir eine Architektur zur Kodierung von AVM-Beschreibungen vorgestellt und skizziert, wie sie für vorhandene HPSG-Werkzeuge nutzbar gemacht werden kann. Die Besonderheiten sind dabei:

- Die DTD ist signatur-spezifisch. Dadurch ermöglichen Standard-XML-Parser bereits einen Syntaxcheck, der feststellt, ob eine XML-Kodierung einer wohlgeformten AVM-Beschreibung entspricht.
- Die AVM-Beschreibungen beinhalten neben Klammerstrukturen auch relationale Ausdrücke und Junktoren. Damit gibt es eine vollständige XML-Entsprechung zu der in der Literatur als lingua franca der HPSG betrachteten AVM-Beschreibungssprache.
- Die Kodierung von Klammerstrukturen ist nicht-rekursiv. Damit ist sie weitgehend unabhängig von Präferenzen der graphischen Darstellung.

Literaturverzeichnis

[1] P. Bonhomme and P. Lopez. TagML, version 0.4, Wed Nov 24 1999. Verfügbar über `www.loria.fr/~lopez/TAG_XML/donnees.html`, 1999.

[2] B. Carpenter and G. Penn. Compiling typed attribute-value logic grammars. In H. Bunt and M. Tomita, editors, *Recent Advances in Parsing Technologies*, pages 145–168. Kluwer, 1996.

[3] D. Flickinger, A. Copestake, and I. A. Sag. HPSG Analysis of English. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, Artificial Intelligence, pages 255–265. Springer-Verlag, Berlin, Heidelberg, New York, 2000.

[4] F. Fouvry and D. Meurers. Towards a platform for linearization grammars. In *On-line Proceedings of the Workshop on Linguistic Theory and Grammar Implementation August 14 - 18, 2000*, 2000. Verfügbar über: `www.sfs.nphil.uni-tuebingen.de/~dm/events/esslli00/proceedings/`.

[5] T. Götz and W. D. Meurers. The importance of being lazy - Using lazy evaluation to process queries to HPSG grammars. In G. Webelhuth, J.-P. Koenig, and A. Kathol, editors, *Lexical and Constructional Aspects of Linguistic Explanation.* CSLI, Stanford, 1998.

[6] E. W. Hinrichs, J. Bartels, Y. Kawata, V. Kordoni, and H. Telljohann. The Tübingen Treebanks for Spoken German, English, and Japanese. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, Artificial Intelligence, pages 552–576. Springer-Verlag, Berlin, Heidelberg, New York, 2000.

[7] E. W. Hinrichs, J. Bartels, Y. Kawata, V. Kordoni, and H. Telljohann. The verbmobil treebanks. In *Proceedings of KONVENS 2000*, 2000.

[8] V. Kordoni. Stylebook for the English Treebank in VERBMOBIL. Technical Report 241, Verbmobil, 2000.

[9] LKB. The (new) LKB system. System entwickelt von Ann Copestake, John Carroll, Rob Malouf, Stefan Oepen und anderen. Dokumentation verfügbar über `www-csli.stanford.edu/~aac/doc5-2a.pdf`, August 2000.

[10] M. Marciniak, A. Mykowiecka, A. Kupść, and A. Przepiórkowski. An HPSG-annotated test suite for Polish. In *Proceedings of the Linguistic Resources and Evaluation Conference*, May 2000.

[11] A. Mengel and W. Lezius. An XML-based encoding format for syntactically annotated corpora. In *Proceedings of LREC 2000*, pages 121–126, Athens, 2000. Verfügbar über: `www.ims.uni-stuttgart.de/projekte/TIGER/paper/lrec200.ps.gz`.

[12] G. Penn. Applying constraint handling rules to HPSG. In *Proceedings of the First International Conference on Computational Logic (CL2000), Workshop on Rule-Based Constraint Reasoning and Programming*, 2000.

[13] C. Pollard and I. A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[14] F. Richter. A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar. Dissertation. Eberhard-Karls-Universität Tübingen, 2000.

[15] H. Wunsch. Grisu: Grammar visualization tool. Unveröffentlichter Kode, 2000. Eberhard-Karls-Universität Tübingen.

[16] XML. Extensible Markup Language (XML) Version 1.0, 1998. Empfehlung, 2. 10. 1998, World Wide Web Consortium, URL: `www.w3.org/TR/1998/REC-xml-19980210`.

[17] XSL. Extensible Stylesheet Language (XSL) Version 1.0, 2000. Working Draft, 27. 3. 2000, World Wide Web Consortium, URL: `www.w3.org/TR/2000/WD-xsl-20000327/`.