Towards an international standard on feature structure representation (2)

Kiyong Lee[1], **Lou Burnard**[2], **Laurent Romary**[3], **Eric de la Clergerie**[4]
Ulrich Schaefer[5], **Thierry Declerck**[6], **Syd Bauman**[7]
Harry Bunt[8], **Lionel Clément**[4]
Tomaz Erjavec[9], **Azim Roussanaly**[3], **Claude Roux**[10]

[1]Korea University Linguistics, Seoul, Korea
klee@korea.ac.kr
[2]Oxford University Computing Services, UK
lou.burnard@oucs.ox.ac.uk
[3]LORIA, France
{Laurent.Romary—Azim.Roussanaly}@loria.fr
[4]INRIA, France
{lionel.clement—Eric.De-La-Clergerie}@inria.fr
[5]DFKI, Germany
ulrich.schaefer@dfki.de
[6]Saarland University & DFKI, Germany
declerck@dfki.de
[7]Brown University, USA
syd_bauman@brown.edu
[8]Tilburg Univeristy, The Netherlands
Harry.Bunt@uvt.nl
[9]Jozef Stefan Institute, Slovenia
tomaz.erjavec@ijs.si
[10]XEROX Research Center Europe, France
claude.roux@xrce.xerox.com

Abstract

This paper describes the preliminary results of a joint initiative of the TEI (Text Encoding Initiative) Consortium and the ISO Committee TC 37SC 4 (Language Resource management) to provide a standard for the representation and interchange of feature structures. The paper published in the proceedings of this workshop is in fact an extension of a paper published in the LREC 2004 proceedings, and about 50% are identical with it.

1. Introduction

This paper describes some preliminary results from a joint initiative of the TEI (Text Encoding Initiative) Consortium and the ISO Committee TC 37/SC 4 (Language Resource management), the goal of which is to define a standard for the representation and interchange of feature structures. The joint working group was established in December 2002, and its proposals are now progressing to Draft International Standard status.

1.1. TEI

Initially launched in 1987, the Text Encoding Initiative (TEI) is an international and interdisciplinary effort the goal of which is to help libraries, publishers, and individual scholars represent all kinds of literary and linguistic texts for online research and teaching, using an encoding scheme that is maximally expressive and minimally obsolescent. The TEI has also played a major role in the development of European language engineering standards since the days of EAGLES. Its recommendations, the "TEI Guidelines", underpin such key standards as the Corpus Encoding Standard, and address many other areas of language resource documentation and description, as well as lexicographic and terminological databases. Since 2000, maintenance and development of the TEI has been managed by an international membership Consortium, which announced publication of a complete XML version of the TEI Guidelines, known as P4 in 2002, and is now overseeing production of a major new revision, known as P5.[1]

1.2. TC 37/SC 4

The research areas of ISO/TC 37/SC 4 include computational linguistics, computerized lexicography, and language engineering. Language resources consist of contents represented by linguistic data in various formats (e.g., speech data, written text corpora, general language lexical corpora). Text corpora, lexica, ontologies and terminologies are typical instances of language resources to be used for language and knowledge engineering. In both monolingual and multilingual environments, language resources play a crucial role in preparing, processing and managing the information and knowledge needed by computers as well as humans. With a view to mobile computing and mobile content etc., the availability of language resources, having to be considered as multilingual, multimedia and

[1]See also http://www.tei-c.org/

multimodal from the outset will be one of the key success factors.[2]

1.3. Current topics of the joint group

The joint TEI and ISO activity has focussed on the following topics:

- articulation of a detailed technical proposal for an XML format able to represent a feature structure analysis with a precise description of the underlying formal mechanism to ensure the coherence and soundness of the standard in line with major theoretical works in this domain;
- provision of specific mechanisms to deal with reentrant structures, clearly distinguished from a generic pointing mechanism;
- provision of a coherent description of the notion of type, which will enable further development of the standard to include a complementary set of proposal relating to declaration of a Feature System.
- integration of this proposal into the on-going revision of the TEI Guidelines (TEI P5) due for publication in 2004;

2. Goal of the paper

The paper first introduces the basic concepts of the features structure formalism. Section 4 briefly describes the proposal currently being developed as an ISO Standard and its relation to other ongoing work relating to the deployment within ISO TC 37 of a general data category registry for linguistic description. The current proposals will include this and other external sources for use, as a reference, in the declaration of particular feature sets. Finally some conclusions are drawn.

3. Feature structures

Feature structures (FSs) form an essential part of many language processing systems, whether their focus is on the description, enrichment, storage, or management of linguistic data. The FS formalism itself has a formal background in graph theory, and supports powerful unification mechanisms for combining elementary structures, which have facilitated its use in many real-world applications. There are many possible ways of representing FSs, but the basic notions have an intrinsic legibility which make them very useful for representing linguistic information in interchange situations, both between people and between processing systems. To take full advantage of this capability, a standard way of representing such structures in electronic format should be made available so that a) specialists from diverse application fields can share detailed expertise from diverse domains and b) implementers can share basic libraries dedicated to the manipulation of FSs, thus reducing the overall cost of application development.

FSs are uniform objects that can be used to represent a wide range of objects, ranging from very simple structures consisting of simple lists of feature-value pairs, to highly complex typed and nested structures with reentrancy, as found for instance in HPSG (Pollard and Sag, 1994b), LFG (Bresnan, 1982), etc. More recently, FSs have also been used as the internal representation for shallow and robust NLP systems based on finite state technologies, or for merging information sources coming from distinct modalities in multi-modal systems.

4. The proposal

This proposal combines a basic set of tags for representing features and feature structures covering in a uniform way the full range of complexity attested by current implementations, together with additional mechanisms to describe libraries of values, feature value pairs and feature structures. As an example, consider the following simple morpho-syntactic annotation for the word ‘vertes’ in French:

```
<fs>
   <f name='token'>
     <string>vertes</string>
   </f>
   <f name='lemma'>
     <string>vert</string>
   </f>
   <f name='pos'>
     <symbol value='adj'/>
   </f>
   <f name='gender'>
     <symbol value='fem'/>
   </f>
   <f name='number'>
     <symbol value='plural'/>
   </f>
</fs>
```

In this XML representation, the element `<fs>` is used to encode a feature structure, and the `<f>` element is used for each of five feature-value pairs making up this structure. Each feature-value pair has a name, given by the name attribute, and contains a primitive or atomic value, marked (in this case) by either a `<string>` or a `<symbol>` element, depending on its datatype. Other possible child elements for the <f> element include <binary> for binary- or boolean-values such as PLUS or MINUS, and <numeric> for various kinds of numeric values and ranges. Complex values can also be represented: collections or multivalues such as lists, sets or multisets (bags) are tagged using a <coll> element; feature structures may also be used as feature-values, thus providing a recursive ability. The components of particular feature structures may be represented directly or referred to by using pointers to previously stored “libraries” of features or feature values. We believe that this XML representation has equivalent expressive power to the classical AVM (Attribute-Value-Matrix) notation, but is more readily processed.

In developing the XML representation, the work group was able to simplify considerably the original TEI proposals as described in (Langendoen and Simons, 1995b), by fo-

[2] See also http://www.tc37/sc4.org/

cussing on applications of the formalism in linguistic analysis alone. The availability of new XML-based tools, in particular the relax-NG schema language now used to express the TEI markup scheme, also proved beneficial for developing a powerful and expressive formalism, adequate to the needs of those using feature structure analysis.

Applications for this formalism have demonstrated the need for more complex mechanisms, which are needed to handle elaborated linguistic information structures. Following on from reference works by Shieber (PATR-II) (Shieber, 1986) or (Carpenter, 1992), there has been a whole range of implementations of FSs in computational linguistics applications. Examples include LOGIN/LIFE (Ait-Kaci and Nasr, 1986), ALE (Carpenter and Penn, 1996), Profit (Erbach, 1995), DyALog (de la Clergerie, 2002), ALEP (Simpkins and Groenendijk, 1994), WAM-like Abstract Machine for TFS (Wintner and Francez, 1995), etc. From another point of view, one can consider the variety of linguistic levels concerned with such representations, e.g. phonology, morpho-syntax, grammars (unification grammars: LFG, HPSG, XTAG), linguistic knowledge base or practical grammar implementation guide (LKB, (Copestake, 2002)), underspecified semantics (MRS, (Copestake et al., 1999)), or integration of NLP components (Schaefer, 2003).

In our work, we have identified and discussed a certain numbers of concepts and topics introduced in the works cited above and we are proposing an XML-based way of representing the corresponding feature structures. As examples, given for this short paper, we show the actual XML implementation of *structure-sharing* (also called *reentrancy*) and the XML treatment of *types*, two topics mentioned in 1.3.:

4.1. Structure Sharing

As shown in most of the works cited above, *structure sharing* (or *reentrancy*) requires the use of labelling for representation in graphic notation such as AVM. For example, to show that a given feature-value pair (or feature structure) occurs at multiple points in an analysis, it is customary to label the first such occurrence, and then to represent subsequent ones by means of the label.

In discussing how to represent this in an XML-based notation, we first proposed making use of a global attribute `label` or `n`, as in the following simple example:

```
<fs>
 <f name="specifier">
  <fs>
    <f name="agr" n="@1">
    <fs>
       <f name="number">
          <symbol value="singular"/>
       </f>
    </fs>
    </f>
    <f name="pos">
         <sym value="determiner"/>
    </f>
  </fs>
 </f>
 <f name="head">
   <fs>
    <f name="agr" n="@1"/>
    <f name="pos">
       <sym value="noun"/>
    </f>
   </fs>
 </f>
</fs>
```

The feature named “agr” is here labelled “@1”. Its first occurrence contains a feature-value pair (“singular number”); its second references this same feature-value pair.

An alternative way of representing this phenomenon is to use the XML ID/IDREf mechanism, as follows:

```
<fs>
 <f name="specifier">
  <fs>
    <f name="agr" id="N1">
    <fs>
       <f name="number">
          <symbol value="singular"/>
       </f>
    </fs>
    </f>
    <f name="pos">
         <sym value="determiner"/>
    </f>
  </fs>
 </f>
 <f name="head">
   <fs>
    <f name="agr" fVal="N1"/>
    <f name="pos">
       <sym value="noun"/>
    </f>
   </fs>
 </f>
</fs>
```

The working group has identified a need to distinguish the case where co-reference implies copying (or transclusion) of shared structures or values, from the case where co-reference simply implies multiple references to the same object, but has not yet reached a resolution as to which of the possible approaches best meets this need.

4.2. Typed Feature Structure

The *typed feature structure* has become a key tool in the linguistic description and implementation of many recent grammar formalisms,

4.2.1. Types

Elements of any domain can be sorted into classes called *types* in a structured way, based on commonalities of their properties. Such linguistic concepts as *phrase*, *word*, *pos* (parts of speech), *noun*, and *verb* may be represented as features in non-typed feature structures. But in typed feature structure particular feature-value pairs may be treated as types.

By *typing*, each feature structure is assigned a particular type. A feature specification with a particular value is then constrained by this typing. A feature structure of the type *noun*, for instance, would not allow a feature like TENSE in it or a specification of its feature CASE with a value of the type *feminine*.[3]

4.2.2. Definition

The extension of non-typed feature structure to typed feature structure is very simple in a set-theoretic framework. The main difference between them is the assignment of types to feature structures. A formal definition of typed feature structure can thus be given as follows:[4]:

Given a finite set of **Features** and a finite set of **Types**, a typed feature structure is a tuple $\mathcal{TFS} = < \mathbf{Nodes}, r, \theta, \delta >$ such that

i. Nodes is a finite set of nodes.

ii. *r* is a unique member of **Nodes** called *the root*.

iii. θ is a total function that maps **Nodes** to **Types**.

iv. δ is a partial function from **Features**$\times$**Nodes** into **Nodes**.

First, each of the **Nodes** must be rooted at or connected back to the root *r*. Secondly, there must one and only one root for each feature structure. Thirdly, each of the **Nodes**, including the root *r* node and terminal nodes, must be assigned a type by the typing function θ. Finally, each of the **Features** labelling each of **Nodes** is assigned a unique value by the feature value function δ.[5]

This type type of information can be encoded in an XML notation, as an example (simplified, due to the length of the paper) shows below:

```
<fs type="word">
  <f name="orth">
    <string>love</string>
    </f>
  <f name="syntax">
    <fs type="verb">
      <f name="valence">
        <symbol value="transitive"/>
      </f>
    </fs>
  </f>
</fs>
```

Note here that the line <f name="pos"><sym value="verb"/></f> in the embedded feature structure <fs> has been replaced by typing that <fs> as in <fs type="verb">.

The use of *type* may also increase the expressive power of a graph notation. On the typed graph notation, for instance, multi-values can be represented as terminating nodes branching out of the node labelled with the type *set*, *multiset* or *list*. This node in turn is a terminating node of the arc labelled with a multit-valued feature, say SLASH. Each arc branching out of the multi-valued node, say *set*, is then labelled with a feature appropriate to the type.

[3]Note that atomic feature values are considered *types*, too.

[4]Slightly modified from (Carpenter, 1992).

[5]The unique-value restriction on features does not exclude multi-values or alternative values because even in these cases each feature ultimately takes a single value which may be considered complex in structure.

4.3. The Equivalence of the XML Representation and the AVM Annotation

The proposed XML representation having equivalent expressive power as the classical AVM notation for feature structures, from a semantic point of view the XML expressions can be interpreted as graphs in the classical way (Carpenter, 1992). In this approach, feature structures are viewed as a *graphs*, i.e., as a certain class of set-theoretical constructs. Carpenter defines a typed feature structure as, given a set Feat of features and a set Type of (hierarchically ordered) types, a quadruple

(1) $< N, n_0, \theta, \mathcal{F} >$

where N is a finite set whose elements are called *nodes*; where $n_0 \in N$, where θ is a total function from N to Type (typing) and where $\mathcal{F}$ is a partial function from $N \times \mathsf{Feat}$ to N (defining arcs, labelled with feature names, that connect the nodes). The node n_0 is the root of the graph; every node in N is required to be reachable from the root node. Pollard and Sag (1987) use this view when they introduce feature structures as semantic entities in the interpretation of representations of linguistic information. They refer to graphs as "modelling structures", i.e., as structures that play a role in models, and they introduce AVMs as structures in a "description language" that is to be interpreted in terms of feature structures-as-graphs: *"Throughout this volume we will describe feature structures using attribute-value (AVM) diagrams"*. (Pollard & Sag, 1987, 19–20).

This view corresponds to the following metamodel that distinguishes nonterminal and terminal nodes and types:

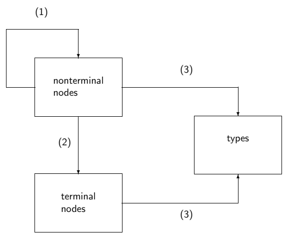

Diagram 1: Metamodel with graphs as model elements

Relations of type (1) in this metamodel correspond to features like HEAD-DAUGHTER in HPSG, those of type (2) to atomic-valued features like GENDER, and those of type (3) to the typing function θ.

An alternative view is that of graphs as *representations*, as a notational alternative to AVMs rather than as the objects interpreting AVMs. For example, Lee (2004) introduces feature structures as ways of capturing information, and mentions graphs as a *notation* for feature structures. Aware of these alternative possible views, Pollard & Sag (1987) note that *"A common source of confusion is that feature structures themselves can be used as descriptions of other feature structures."* One way to avoid confusion is to consider the metamodels corresponding to alternative views.

In the graphs-as-representations view, the graph (2) and the AVM (3) are seen as equivalent representations that can both be interpreted as representing the complex predicate (4).

(2)
$$\begin{tikzcd}
\text{noun} \arrow[r, "\text{AGR}"] & \text{agr} \arrow[r, "\text{NUM}"] \arrow[dr, "\text{GENDER}"'] & \text{sing} \\
 & & \text{fem}
\end{tikzcd}$$

(3)
$$\begin{bmatrix} \textit{noun} & \\ \text{AGR} & \begin{bmatrix} \text{NUM sing} \\ \text{GENDER fem} \end{bmatrix} \end{bmatrix}$$

(4) $\lambda x : noun(x) \wedge num(x) = sing \wedge gender(x) = fem$

(simplifying slightly). This interpretation reflects a similar view on information as that of first-order logic, with two kinds of individuals: the kind of things that x stands for (words and phrases) and the kind of atomic attribute values like ‘fem’ and ‘sing’. These values are associated with word-like individuals through two-place predicates that are in fact functions; moreover, types such as ‘noun’ correspond to unary predicates. This corresponds to the metamodel visualized in Diagram 2.

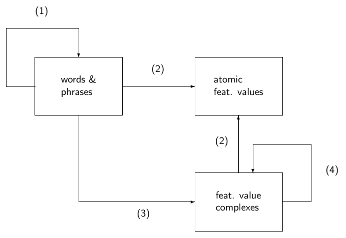

Diagram 2: First-order metamodel for feature structures

Relations of type (1) in this diagram (1) correspond again to features like HEAD-DAUGHTER; (2) to atomic-valued features like GENDER; (3) to features like SYNSEM, and (4) to features like AGR(EEMENT).

5. The role of feature structure markup transformation for the integration of NLP components

One of the main motivations for XML feature structure markup is the interchange of linguistic data. This can be done *offline*, e.g., for the exchange of lexica, grammatical resources, or annotated documents.

A further application is *online* integration of NLP components, where several, specialised modules contribute to improved (e.g., disambiguated or more precise) linguistic analyses. Examples for such hybrid architectures are Whiteboard (Frank et al., 2003; Schäfer, 2003) and DeepThought (Callmeier et al., 2004).

In both cases, online or offline integration, different representations of linguistic data can be involved, where feature structures can either form the source or the target representation or even both.

In general, conversion or translation of different XML representations is required. In the case of XML, such a translation is called transformation, and the established W3C standard language for XML transformation is XSLT (eXtensible Styleheet Transformation; (Clark, 1999)).

The input of an XSL transformation is always XML, while the output can be of any syntax, including XML as a well-supported target format.

To illustrate the use of XML transformation for of feature structure markup, we give concrete examples.

5.1. Feature structures as target representation

Construction of (typed) feature structures from other XML representations that are e.g. produced by a shallow NLP system. Specific elements with attributes are translated to possibly nested feature-value pairs, e.g. for input to a HPSG(Pollard and Sag, 1994a) parser etc. In the following example, `<infl num="singular"/>` is translated to the corresponding feature structure. Of course, also symbolic names e.g. sg to singular etc. can be translated.

```
<xsl:template match="infl">
   <fs type="infl">
       <f name="number">
          <symbol value="@num"/>
       </f>
   </fs>
</xsl:template>
```

Grammar exchange format or meta syntax like in SProUT (Drozdzynski et al., 2004), where a TDL-like grammar syntax (Krieger and Schäfer, 1994) is translated to an internal representation based on feature structure XML. The internal representation is used as input for type checking and compilation.

Data exchange between NLP components, e.g. the so-called SProUTput DTD that is used for exchange of typed feature structures with external NLP components (input and output) in SProUT[6].

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- SProUTput DTD (2003) -->
<!ELEMENT SPROUTPUT ( DISJ )* >
<!ELEMENT DISJ ( MATCHINFO )+ >
<!ATTLIST DISJ id ID >
<!ELEMENT MATCHINFO ( FS ) >
<!ATTLIST MATCHINFO id ID      #IMPLIED
                  rule NMTOKEN #IMPLIED
                cstart NMTOKEN #IMPLIED
                  cend NMTOKEN #IMPLIED
                 start NMTOKEN #IMPLIED
                   end NMTOKEN #IMPLIED >
<!ELEMENT FS ( F )* >
<!ATTLIST FS      type NMTOKEN #REQUIRED
                 coref NMTOKEN #IMPLIED >
<!ELEMENT F  ( FS ) >
<!ATTLIST F       name NMTOKEN #REQUIRED >
```

5.2. Feature structures as source representation

Extraction or projection of information encoded in feature structures such as morphology to other formats or as API-like accessors. e.g. an XPath expression like

[6]Element names are uppercase in the SProUTput DTD

```
  <xsl:template match="fs[@type='infl']">
    <infl num="f[@name='number']/symbol
                                /@value"/>
  </xsl:template>
```

is the inverse of example 5.1. above.

AVM visualisation tools or editors like the feature structure renderer in SProUT or Thistle (Calder, 2000) both take (different) descriptions of typed feature structures and render a graphical representation of the feature structure.

Extraction of tree structures etc. from a a complex HPSG feature structure, e.g. for further linguistic processing or visualisation in Thistle.

Generation of semantics representation. An example is a transformation of typed feature structures to RMRS XML markup (Copestake, 2003) which e.g. forms the basic representation for the exchange of deep and shallow NLP processing results in the DeepThought architecture (Callmeier et al., 2004), cf. Fig. 1.

5.3. Feature structures as both source and target representation

Translation between different feature structure syntaxes or systems. We give an example of list values that can be encoded differently in typed feature structure markup. The XSLT template below takes a list encoded as nested `FIRST-REST` list typed `*cons*` and translates it to the proposed `<list>` with embedded elements from the `FIRST` attribute values in the input. The template works recursively on `FIRST-REST` lists of any length.

```
<!-- ======================================
Initial template. Enclose list elements from
FIRST-REST list in <list> element
======================================= -->
<xsl:template match='fs[@type="*cons*"]'>
 <xsl:element name="list">
  <xsl:call-template name="listlist">
   <xsl:with-param name="node" select="."/>
  </xsl:call-template>
 </xsl:element>
</xsl:template>
<!-- ======================================
recursive template: list all list elements
======================================= -->
<xsl:template name="listlist">
 <xsl:param name="node"/>
 <xsl:copy-of select='$node/f[@name=
                           "FIRST"]/fs'/>
 <xsl:if test='$node/f[@name="REST"]/fs/
                         @type="*cons*"'>
  <xsl:call-template name="listlist">
   <xsl:with-param name="node"
        select='$node/f[@name="REST"]/fs'/>
  </xsl:call-template>
 </xsl:if>
</xsl:template>
```

5.4. Reentrancies and Transformation

A general issue that arises for the case where feature structures are source representations is reentrancies. Here, ‘dereferencing‘ is necessary on the basis of lookup in the XML source in order to have access to every node in the DAG (e.g. for feature path access); XML ID/IDREF declarations support faster access as discussed already before. If cyclic reentrancies are disallowed, copying of shared values when generating the features structure representation is an easy and probably quicker way in order to get the full access to shared values. Identity information is preserved through the reentrancy attribute anyway.

6. Related work within the ISO framework

A distinctive feature of the TEI Guidelines is its use of an integrated model of documentation and documentation outputs. The ODD system used to produce its recommendations, both as printed documentation and as formal syntax expressed in XML Schema or DTD languages, has recently been revised and re-expressed. This new modular system for documentation is likely to have wide take up in many different domains. In applying it to the expression of the feature structure analysis language, we have identified a number of potential areas of synergy with the ongoing ISO work on data category registry[7].

7. Conclusions

The work reported has proved to be an excellent opportunity for experimenting with the new descriptive framework being developed for the TEI Guidelines themselves. The feature structure activity has been a useful opportunity to experiment with the creation of relevant tagging systems and tools in a relatively limited but formally complex domain.

In general, the activity reported in the paper shows that there is great scope for further convergence between the TEI consortium and ISO committee TC 37/SC 4, and many benefits to be gained from joint work on issues which require complementary expertise in textual representation methods and in the representation of linguistic concepts.

[7]See for more details: http://jtc1sc36.org/doc/36N0581.pdf

```
<MATCHINFO rule="en_city" cstart="3" cend="7">
  <FS type="sprout_rule">
    <F name="OUT">
      <FS type="ne-location">
        <F name="LOCNAME">
          <FS type="&quot;Paris&quot;"/>
        </F>
        <F name="LOCTYPE">
          <FS type="city"/>
        </F>
      </FS>
    </F>
  </FS>
</MATCHINFO>
```

```
    <rmrs cfrom="3" cto="7">
      <label vid="1"/>
      <ep cfrom="3" cto="7">
        <gpred>ne-location</gpred>
        <label vid="2"/>
        <var sort="x" vid="2"/>
-->   </ep>
      <rarg>
        <label vid="2"/>
        <rargname>CARG</rargname>
        <constant>"Paris"</constant>
      </rarg>
    </rmrs>
```

Figure 1: Transformation of feature structure XML markup (SProUT) to RMRS (DeepThought).

8. References

Ait-Kaci, H and R Nasr, 1986. Login: A logic programming language with built-in inheritance. *J. Log. Program.*, 3(3):185–215.

Bering, Christian, Witold Drozdzyski, Gregor Erbach, Clara Guasch, Petr Homola, Sabine Lehmann, Hong Li, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Atsuko Shimada, Melanie Siegel, Feiyu Xu, and Dorothee Ziegler-Eisele, 2003. Corpora and evaluation tools for multilingual named entity grammar development. In *Proceedings of Multilingual Corpora Workshop at Corpus Linguistics 2003*. Lancaster.

Bresnan, Joan (ed.), 1982. *The Mental Representation of Grammatical Relations*. Cambridge, MA: The MIT Press.

Busemann, Stephan, Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Hans Uszkoreit, and Feiyu Xu, 2003. Integrating information extraction and automatic hyperlinking. In *Proceedings of ACL-2003, Interactive Posters/Demonstrations*. Sapporo, Japan.

Calder, Joe, 2000. *Thistle: Diagram Display Engines and Editors*. HCRC, U. of Edinburgh.

Callmeier, Ulrich, 2000. PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6 (1):99 – 108.

Callmeier, Ulrich, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel, 2004. The DeepThought core architecture framework. In *Proceedings of LREC-2004*. Lissabon, Portugal.

Carpenter, Bob, 1992. *The Logic of Typed Feature Structures*. Cambridge University Press.

Carpenter, Bob and Gerald Penn, 1996. Efficient parsing of compiled typed attribute value logic grammars. In H. Bunt and M. Tomita (eds.), *Recent Advances in Parsing Technology*. Kluwer, page Recent Advances in Parsing Technology.

Clark, James, 1999. *XSL Transformations (XSLT)*. W3C, http://w3c.org/TR/xslt.

Clark, James and Steve DeRose, 1999. *XML Path Language (XPath)*. W3C, http://w3c.org/TR/xpath.

Copestake, Ann, 2002. *Implementing Typed Feature Structure Grammars*. Stanford, CA: CSLI publications.

Copestake, Ann, 2003. Report on the design of RMRS. Technical Report D1.1b, University of Cambridge, Cambridge, UK.

Copestake, Ann, Dan Flickinger, Ivan Sag, and Carl Pollard, 1999. Minimal recursion semantics: An introduction. Draft.

Crysmann, Berthold, 2003. On the efficient implementation of german verb placement in HPSG. In *Proceedings of RANLP-2003*. Borovets, Bulgaria.

Crysmann, Berthold, Anette Frank, Bernd Kiefer, Stefan Müller, Jakub Piskorski, Ulrich Schäfer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker, and Hans-Ulrich Krieger, 2002. An Integrated Architecture for Deep and Shallow Processing. In *Proceedings of ACL 2002*. Philadelphia, PA.

de la Clergerie, Éric Villemonte, 2002. Construire des analyseurs avec dyalog. In *Proceedings of TALN '02*.

Drozdzynski, Witold, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu, 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23. Http://www.kuenstliche-intelligenz.de/archiv/2004_1/sprout-web.pdf.

Erbach, Gregor, 1995. Profit: Prolog with features, inheritance, and templates. In *Proceedings of EACL '95*.

Frank, Anette, Markus Becker, Berthold Crysmann, Bernd Kiefer, and Ulrich Schäfer, 2003. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of ACL-2003*. Sapporo, Japan.

Kasper, Walter, Jörg Steffen, Jakub Piskorski, and Paul Buitelaar, 2004. Integrated language technologies for multilingual information services in the MEMPHIS project. In *Proceedings of LREC-2004*. Lissabon, Portugal.

Krieger, Hans-Ulrich and Ulrich Schäfer, 1994. *TDL*— a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*.

Langendoen, D. Terence and Gary F. Simons, 1995a. A rationale for the TEI recommendations for feature structure markup. In Nancy Ide and Jean Veronis (eds.), *Computers and the Humanities 29(3)*. The Text Encoding Initiative: Background and Context, Dordrecht: Kluwer Acad. Publ. Reprint.

Langendoen, Terence D. and Gary F. Simons, 1995b. A rationale for the tei recommendations for feature-structure markup. *Computers and the Humanities*, 29:191–209.

Lee, Kyiong, 2004. Language resource management – feature structures part1: Feature structure representation. Document iso/tc 37/sc 4 n 033, ISO.

Pollard, Carl and Ivan A. Sag, 1994a. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Chicago: University of Chicago Press.

Pollard, Carl J. and Ivan A. Sag, 1987. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Pollard, Carl J. and Ivan A. Sag, 1994b. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Sailer, Manfred and Frank Richter, 2001. Eine XML-Kodierung für AVM-Beschreibungen. In Henning Lobin (ed.), *Proceedings der GLDV-Frühjahrstagung 2001*. Gesellschaft für linguistische Datenverarbeitung.

Schaefer, Ulrich, 2003. What: An xslt-based infrastructure for the integration of natural language processing components. In *Proceedings of HLT-NAACL 2003 Workshop: Software Engineering and Architecture of Language Technology Systems*.

Schäfer, Ulrich, 2003. WHAT: An XSLT-based Infrastructure for the Integration of Natural Language Processing Components. In *Proc. of the Workshop on the Software Engineering and Architecture of LT Systems (SEALTS), HLT-NAACL03*. Edmonton, Canada.

Shieber, Stuart M., 1986. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes Series*. Stanford, CA: Center for the Study of Language and Information.

Simpkins, N. and M. Groenendijk, 1994. The alep project. technical report. Technical report, Cray Systems / CEC.

Thompson, Henry S. and David McKelvie, 1997. Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML-EU-1997*.

Uszkoreit, Hans, 2002. New Chances for Deep Linguistic Processing. In *Proceedings of COLING 2002*. Taipei, Taiwan.

Wintner, Shuly and Nissim Francez, 1995. Parsing with typed feature structures. In *Proceedings of the Fourth International Workshop on Parsing Technologies*.