Standards for the Formal Representation of Linguistic Data: An Exchange Format for Feature Structures

Rainer Osswald
SFB 991
Heinrich-Heine-Universität Düsseldorf
osswald@phil.hhu.de

Abstract

The International Standard ISO 24610 defines a schema of how to encode feature structures and their declarations in XML. The main goal of this standard is to provide a format for the exchange of feature structures and feature system declarations between applications. This paper gives an overview of the elements of the standard and sketches its development and some of the design decisions involved. We also discuss the role of this standard in relation to other standardization proposals for language resources and we briefly address its relevance for application programming.

1 Feature structures

1.1 Feature structures in linguistics

In modern linguistics, the characterization of linguistic entities as bundles of *distinctive features* has been employed most prominently in phonology.[1] The phoneme /p/, for instance, can be seen as the result of combining the phonetic features *voiceless*, *plosive*, *bilabial*, etc. Phonetic features occur usually in opposition pairs such as *voiced* vs. *voiceless* and *plosive* vs. *non-plosive*. This dichotomy can be taken into account by introducing *binary* features like VOICE, PLOSIVE, etc., with possible values + and −. The phoneme /p/ is then described by a set of feature-value pairs, for which the following matrix notation is in use:

$$\begin{bmatrix} \text{VOICE} & - \\ \text{PLOSIVE} & + \\ & \vdots \end{bmatrix}$$

[1] Cf. Chomsky and Halle (1968).

$$\begin{bmatrix} \text{CATEGORY} & \textit{noun} \\ \text{WORDFORM} & \text{'Junge'} \\ \text{AGREEMENT} & \begin{bmatrix} \text{GENDER} & \textit{masculine} \\ \text{NUMBER} & \textit{singular} \\ \text{CASE} & \textit{nominative} \end{bmatrix} \end{bmatrix}$$

Figure 1: Example of an untyped feature structure.

With the advent of unification-based grammars in (computational) linguistics such as Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985) and Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), more complex, nested feature structures arose. Figure 1 shows a simple example of a nested feature structure, in which the non-atomic value of the feature AGREEMENT is again a feature structure. The example describes part of the grammatical information associated with the German noun 'Junge' ('boy'). The feature structure is *untyped* in that neither the main matrix nor the embedded matrix carries any sortal information. In this respect, this example differs from the *typed* feature structure shown in Figure 2, in which the main structure and every substructure carry a type (*phrase*, *word*, etc.). The latter example also illustrates the notion of *structure sharing*. The values of the AGREEMENT feature are (token) identical, which indicates that in a noun phrase, the agreement features of the determiner and those of the head noun must coincide.

The International Standard ISO 24610 provides a schema for the representation of feature structures in XML. The main purpose of such a representation is the standardized exchange of data be-

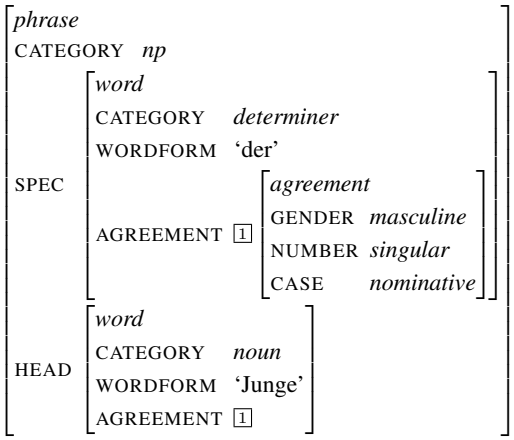

Figure 2: Example of a typed feature structure with structure sharing.

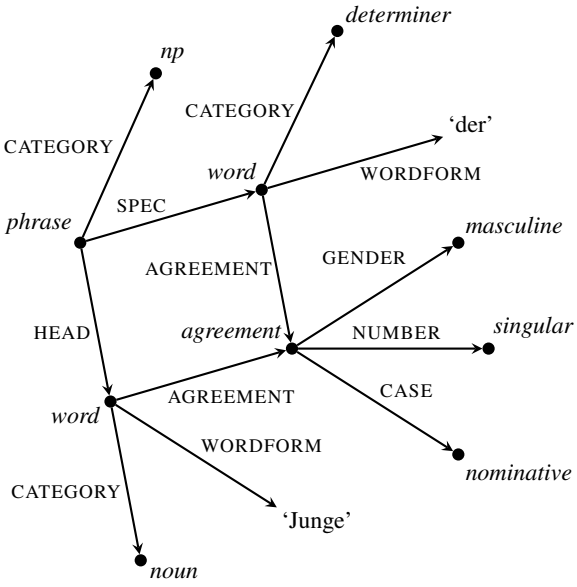

Figure 3: Labeled directed graph representation of the feature structure of Figure 2.

tween different applications. In what follows, we first give a formal definition of feature structures. Section 2 is concerned with the rationale behind the XML representation defined by the standard, while Section 3 gives an overview of the elements of the standard itself. In the concluding Section 4, we will briefly discuss how this standard is related to other standardization proposals for language resources and what role the standard can play for application programming.

1.2 Formal definition of feature structures

In Part 2 of the International Standard ISO 24610, feature structures are formally defined as follows. The definition presumes a given finite set $\mathcal{F}$ of features, a finite type hierarchy $\mathcal{T}$ with subtyping relation $<$ and a set $\mathcal{X}$ of "built-in" elements (see below).[2]

Definition A *feature structure* over $\mathcal{F}$, $\mathcal{T}$ and $\mathcal{X}$ is a quadruple $\langle Q, \rho, \theta, \delta \rangle$, in which

- Q is a set of *nodes*,
- ρ is an element of Q, called the *root*,
- θ is a partial *typing function* from Q to $\mathcal{T}$,
- δ is a partial *feature value function* from $\mathcal{F} \times Q$ to $Q \cup \mathcal{X}$,

such that, for every node $q \neq \rho$, there exists a sequence $f_1, \ldots, f_n$ of features and a sequence $q_1, \ldots, q_n$ of nodes with $q_1 = \rho$, $q_n = q$ and $q_{i+1} = \delta(f_i, q_i)$ for $i < n$.

[2] A type hierarchy is a finite ordered set $\langle \mathcal{T}, < \rangle$ such that every two elements of $\mathcal{T}$ have a least upper bound in $\mathcal{T}$. In particular, every type hierarchy has a greatest element.

The last condition of the definition basically says that every node of a feature structure can be reached from the root by a feature sequence, or *path*. The given definition slightly generalizes the definitions typically found in the literature (e.g., Carpenter (1992)) in that, first, the typing function is only partial and thus allows for untyped feature structures and, second, the value of a feature can belong to a set $\mathcal{X}$ of elements defined elsewhere. The members of $\mathcal{X}$ can be thought of as built-ins such as binary values, string values, symbolic values and numeric values.

The relation between the above definition of feature structures and the feature-value matrices shown before is fairly straightforward. Roughly speaking, every opening square bracket and every atomic feature value (which is not a built-in) corresponds to a node. The typing function is determined by the types attached to the matrix brackets and by the atomic values (except, again, for built-ins). The feature value function is defined in accordance with how the features in the matrix connect one bracket to another or to an atomic value.[3]

[3] There has been some discussion in the literature (e.g. Carpenter (1992), Pollard and Sag (1994)) about the question as to whether feature-value matrices should better be seen as *descriptions* of feature structures. This is not the place to take up this discussion, but it seems that if this dis-

Figure 3 depicts the feature structure of Figure 2 as a labeled directed graph, where the labeled edges represent the feature value function and the labels of the nodes represent the typing function. Note that the two string values are treated as built-ins in this example.

2 XML representation of feature structures

2.1 Data-oriented XML

The Extensible Markup Language (XML) is the most popular format for exchanging structured data and thus a natural candidate for providing an interoperable representation format for feature structures.[4] Since XML documents have an associated tree model (and even provide means for expressing co-reference), they seem to be fairly closely related to feature structures.

There are, however, crucial differences between the two structures, which requires various design decisions concerning the precise representation of feature structures by means of XML. First, XML document trees consist of categorically different types of nodes, namely element nodes, attribute nodes, text nodes and some others, of which only element nodes can have descendants (keeping aside the case of 'mixed content').[5] A second difference is that XML trees are *ordered* trees (like any representation that is based on serialization), in contrast to features structures. The inherent ordering of XML provides an easy way to encode list values (see Section 3.1) but, on the other hand, gives rise to an identity problem since the order of the features matters for the XML representations but not for represented feature structures. This issue could be resolved, if necessary, by a canonical, say, alphabetic ordering of the features.

Another important aspect of an XML representation is that it is well-suited for declaring and checking its structure by available XML schema languages and tools.

2.2 A rationale for feature structure representation in XML

Many of the design decisions that led to the ISO 24610 standard can already be found in the TEI Guidelines P3 dating back to the mid-1990's, with SGML instead of XML in use at that time. Langendoen and Simons (1995) give a concise exposition of the decision process, which we briefly summarize in the following. First, it is clear that representing features by XML attributes is no option since attributes cannot have descendants and thus would not allow to represent nested feature structures. The following option is more promising: It seems fairly natural to represent features by XML elements. The feature structure shown in Figure 1 could then be represented in XML as follows:

```
<fs>
  <category>noun</category>
  <wordform>Junge</wordform>
  <agreement>
    <fs>
      <gender>masculine</gender>
      <number>singular</number>
      <case>nominative</case>
    </fs>
  </agreement>
</fs>
```

As Langendoen and Simons (1995) point out, the main disadvantage of this representation is the unrestricted proliferation of elements. As a consequence, there would be no way to define a general schema for feature structure representations since the content model would depend fully on application-specific elements. The proposed solution is to employ a general-purpose element `f` for features and to represent the names of the features as attributes of these elements:

```
<fs>
  <f name="category">noun</f>
  <f name="wordform">Junge</f>
  <f name="agreement">
    <fs>
      <f name="gender">masculine</f>
      <f name="number">singular</f>
      <f name="case">nominative</f>
    </fs>
  </f>
</fs>
```

tinction is relevant at all then an XML-based interchange format is more on the description side of the divide.

[4] Some basic knowledge of XML is assumed in the following; cf. Ray (2003) for an introduction.

[5] An XML document is said to have *mixed content* if it allows for elements which may contain character data that are interspersed with child elements. This is typically the case in text-oriented applications such as DocBook. Mixed content is generally regarded as inappropriate for data-oriented formats like the ones discussed in this article.

The representation of the feature values poses a second issue. The previous proposal does not distinguish between arbitrary strings such as 'Junge' and symbols such as *noun* and *singular*, which belong to a given set of symbols. Moreover, the content model for `f` would allow text content as well as structured content. These problems can be resolved by embedding the feature values in separate elements such as `symbol` and `string`. String values are then encoded as text content of `string` and type symbols are represented as values of the attribute `value` of the element `symbol`, in line with the general practice to use text content in data-oriented XML for unrestricted text only. The resulting XML representation of the above feature structure now looks as follows:

```
<fs>
  <f name="category">
    <symbol value="noun"/>
  </f>
  <f name="wordform">
    <string>Junge</string>
  </f>
  <f name="agreement">
    <fs>
      <f name="gender">
        <symbol value="masculine"/>
      </f>
      <f name="number">
        <symbol value="singular"/>
      </f>
      <f name="case">
        <symbol value="nominative"/>
      </f>
    </fs>
  </f>
</fs>
```

Note that the typing of feature structures is not part of the above schema but could easily be implemented by an appropriate attribute of the `fs` element; cf. Section 3.1 below.

The structural schema of the feature structure representation developed so far is summarized by the following RELAX NG schema:[6]

```
fs =
  element fs {
    element f {
      attribute name { text },
      ( symbol | string | fs )
    }*
  }

symbol =
  element symbol {
    attribute name { text }
  }

string = element string { text }
```

The proposed schema does not impose any constraints on the features or the values encoded by a feature structure representation. It characterizes the generic structure of feature structure representations in XML. The schema does not even exclude a feature name to occur more than once in the `f` children of an `fs` element. That is, the fundamental property of feature structures that features are functional is not part of the well-formedness of the representations licensed by the above schema. There is also no way to customize the schema in such a way that it would capture more specific constraints about the feature architecture and the type system. Grammar-based schema languages like RELAX NG simply do not support the formulation of conditions between attributes of different elements.[7]

For this reason, Langendoen and Simons (1995, p. 202) argue for devising a separate markup format which is capable of representing all kinds of constraints on the type system and the feature architecture associated with a particular application domain. The resulting XML document is called a *feature system declaration*. It specifies, among others, which features are admissible for a feature structure of a certain type and which values are admissible for a given feature. A further kind of constraint that can be expressed by an FSD is the feature co-occurrence restriction used with untyped feature structures.

3 The International Standard ISO 24610

The International Standard ISO 24610 grew out of a joint initiative of the Text Encoding Initiative

[6] 'RELAX NG' stands for 'REgular LAnguage for XML, New Generation'. The presentations given in this article use the so-called compact syntax of RELAX NG. There is also an XML-based syntax and there are tools to convert RELAX NG schemas into other schema languages such as W3C XML Schema. Cf. van der Vlist (2003) for an excellent introduction to RELAX NG.

[7] The *rule-based* schema language Schematron, by comparison, is able to assert such constraints since it has full XPath support.

```
fs =
  element fs {
    attribute type { xsd:Name }?,
    element f {
      attribute name { text },
      model.featureVal*
    }*
  }

model.featureVal.complex =
  model.featureVal.complex |
  model.featureVal.single

model.featureVal.complex =
  fs | vColl | vNot | vMerge

model.featureVal.single =
  binary | symbol | numeric |
  string | vLabel | default | vAlt

vColl =
  element vColl {
    attribute org { "set" | "bag" |
                    "list" }?,
    ( fs | model.featureVal.single )*
  }

vLabel =
  element vLabel {
    attribute name { data.word },
    model.featureVal?
  }
```

Figure 4: Slightly simplified excerpt of the ISO/TEI XML schema for feature structure representations (FSD).

(TEI) Consortium and the ISO Sub-Committee TC 37/SC 4 (Language Resources Management); cf. Lee et al. (2004). The goal was to develop an international standard for the representation of feature structures which can serve as an exchange format between applications.

The standard consists of two parts. The first part, ISO 24610-1 on features structure representation (FSR), was published 2006 while the second part, ISO 24610-2 on features system declaration (FSD), was published more than five years later. Due to the fairly long time span between the two publication dates and the strong interdependency of the two parts, there are plans for revising the ISO 24610-1 standard in order to make it fully compliant with the more recent additions of Part 2.

3.1 Feature structure representation (FSR)

The ISO standard builds on the schema motivated in Section 2.2. Figure 4 shows a slightly simplified excerpt of the XML schema listed as an normative appendix of the second part of the ISO 24610 document. The main differences compared to the schema developed above is an optional `fs` attribute `type`, which allows for the representation of typed feature structures, and an extended list of elements for representing feature values.

Structure sharing. Let us first consider the element `vLabel`. According to its specification shown in Figure 4, it can serve as a kind of "value wrapper". Its purpose is to allow the representation of structure sharing, with its `name` attribute acting as the co-reference label. For example, the structure sharing between the two *agreement* substructures of the feature structure shown in Figure 2 could be represented as follows:

```
<fs type="np">
  ...
  <f name="spec">
    <fs type="word">
      <f name="agreement">
        <vLabel name="L1">
          <fs type="agreement">
            ...
          </fs>
        </vLabel>
      </f>
      ...
    </fs>
  </f>
  <f name="head">
    <fs type="word">
      <f name="agreement">
        <vLabel name="L1"/>
      </f>
      ...
    </fs>
  </f>
</fs>
```

Built-in value elements. In addition to the already mentioned built-in value elements `symbol` and `string`, the ISO standard defines two further elements: `binary` and `numeric`, which have their obvious intended interpretation.

The `vColl` element allows the encoding of lists, sets and bags (i.e., multisets) of atomic and complex values. The corresponding schema in Figure 4 shows that the members of the collection are represented as children of the `vColl` el-

ement, while the interpretation as a set, list, or bag is simply indicated by the value of the attribute `org`. Note that lists can be straightforwardly represented this way because XML document trees are ordered trees by definition. There is of course no need to make use of the `<vColl org="list">` construct since lists can be represented directly as recursively nested feature structures by means of the two features FIRST and REST.[8]

The `vNot` and `vAlt` elements do not act as constructors but characterize a single value. In the case of `vAlt`, the value is specified as being one of the values listed as children of the element; in the case of `vNot`, the value is specified as being not the value given by the single child of that element.

Symbolic values vs. types. An issue addressed in Part 2 of the ISO 24510 document but not in the TEI P5 Guidelines is the relation between symbolic feature values and types. Starting with untyped feature structures, as we did in Section 2.2, it seems quite natural to introduce an element like `symbol` for encoding symbolic values. In the presence of types, on the other hand, such elements are redundant at best. A symbolic value can be treated as a type, which in turn can be identified with an atomic typed feature structure, that is, a feature structure without any features. The resulting difference in representation is illustrated by the following two examples:

```
<fs type="word">
  <f name="category">
    <symbol value="noun"/>
  </f>
  ...

<fs type="word">
  <f name="category">
    <fs type="noun"/>
  </f>
  ...
```

The advantage of the second representation is that symbolic values become on a par with types and are hence part of the type hierarchy, which is not the case with built-ins (cf. Section 1.2). We will return to this issue below in Section 3.2.

[8] Cf. Witt et al. (2009) for an application of the FSR schema that makes use of the latter option.

```
element fsdDecl { fsDecl+ }

fsDecl =
  element fsDecl {
    attribute type { xsd:Name },
    attribute baseTypes {
      list { xsd:Name+ }
    }?,
    fDecl+, fsConstraints?
  }

fDecl =
  element fDecl {
    attribute name { xsd:Name },
    vRange, vDefault?
  }

fsConstraints =
  element fsConstraints {
    (cond | bicond)*
  }
```

Figure 5: Simplified excerpt of the ISO/TEI XML schema for feature system declarations.

Feature libraries and feature-value libraries. The FSR standard provides means for defining libraries of feature-value combinations and of feature values, collected under the elements `fLib` and `fvLib`, respectively. The idea is that referencing the items of these collections by unique identifiers may result in a more compact representation of a feature structure. Notice that type systems provide an alternative way of defining complex feature-value combinations that can be re-used at different places in other feature structures.

3.2 Feature system declaration (FSD)

Well-formedness vs. validity. As explained in Section 2.2, well-formedness with respect to the FSR schema is not concerned with any specific constraints on the feature architecture or the type system of the represented feature structure. The solution proposed in the TEI Guidelines starting with version P3 was to introduce a separate XML format for specifying such declarations. This idea and its current implementation in the TEI P5 Guidelines have been integrated into Part 2 of the ISO 24610 standard.

Document structure. The overall document structure of a feature system declaration (FSD) is

roughly described by the schema shown in Figure 5. A feature system declaration (`fsdDecl`) consists of one or more feature structure declarations (`fsDecl`). A feature structure declaration specifies the type of the structure and, optionally, one or more "base" types of which the type is a sub-type. In addition, a feature structure declaration consists of one or more feature declarations (`fDecl`), which specify the name of the features, the range of possible values and, optionally, a default value.

Feature structure and type declarations. The following example sketches part of an FSD for the typed feature structure shown in Figure 2. The attribute `baseTypes` is used to declare that the type *word* is a subtype of the type *sign*.

```
<fsDecl type="word" baseTypes="sign">
  <fDecl name="category">
    <vRange>
      <vAlt>
        <symbol value="determiner"/>
        <symbol value="noun"/>
        <symbol value="verb"/>
        ...
      </vAlt>
    </vRange>
  </fDecl>
  <fDecl name="wordform">
    <vRange>
      <string/>
    </vRange>
  </fDecl>
  ...
</fsDecl>
```

The example illustrates how to declare the range of possible feature values by means of the `vRange` construct. In fact, this is the only option if atomic values are represented by built-in elements such as `symbol`. In order to represent *determiner*, *noun*, *verb* etc. as members of the type hierarchy, they have to be treated as atomic feature structures along the following lines:

```
<fsDecl type="word" baseTypes="sign">
  <fDecl name="category">
    <vRange>
      <fs type="category">
    </vRange>
  </fDecl>
  ...
</fsDecl>
```

```
<fsDecl type="determiner"
        baseTypes="category"/>
<fsDecl type="noun"
        baseTypes="category"/>
<fsDecl type="verb"
        baseTypes="category"/>
...
```

Hence, the type hierarchy can be represented by an FSD document by declaring types as atomic feature structures together with the more general types from which the type in question inherits.[9]

Feature structure constraints. One of the goals of the ISO standard is to allow for both, the feature structure declarations of typed feature structures in the style of HPSG (Pollard and Sag, 1994) and the representation of feature co-occurrence restrictions and defaults as employed in GPSG (Gazdar et al., 1985) and other untyped frameworks. While implicational constraints are also relevant for typed frameworks (cf., e.g., the principles of HPSG), they are fundamental to the untyped case. We will not go into detail here because the representation of such constraints is basically a question of how to represent logical expressions in general (which in turn is the topic of other initiatives such as RuleML). Examples adapted from Gazdar et al. (1985) can be found in Burnard and Bauman (2012, Sect. 18.11.4).

4 Discussion

Is FSR/FSD more than an exchange format? The primary use case of the ISO 24610 standard is the exchange of feature structure data between applications. The question arises whether the XML representation of feature structures compliant with FSR is not only useful for exchanging data but also as a data format for application programming. In other words, can the XML representation serve as the native data format of an application? The appropriate answer is a qualified yes. On the one hand, existing XML technology (XML query languages, native XML databases) provides a powerful environment for working with FSR compliant data. On the other

[9] There is a caveat here. The schema for `fsDecl` in ISO 24610-2 and TEI P5 apparently poses a problem for this strategy since it requires at least one `fDecl` child. Hence, strictly speaking, atomic feature structures cannot be declared.

hand, checking the validity of FSR data directly on the basis of an FSD document is probably a tedious task, for which logic programming of one kind or another seems to be much better suited.

Is there a need for software tools? As an application-oriented exchange format, the ISO standard does not aim at human readability. The question is, whether there is a need for viewing and editing or even validity checking of FSR compliant data. Given that an application system typically uses its own internal representation, which is part of an elaborate editing and programming environment, the answer could be negative. However, if the ISO standard is established as an export format, then a sufficiently powerful, freely available (open source) tool for browsing, editing and probably checking FSR/FSD data would be extremely helpful for accessing the data. Moreover, the existence of such a tool could boost the dissemination of the standard. Potential users would need to transform their data into the ISO format in order to use the tool.

The ISO 24610 standard in context. How does the ISO 24610 standard relate to other standards for language resources developed by the ISO Sub-Committee TC 37/SC 4? According to the International Standard ISO 24612 'Linguistic annotation framework' (LAF), the FSR standard plays a key role for any sort of annotation. LAF draws a clear-cut distinction between *referential structure* and *annotation content structure* and proposes that the latter be represented as feature structures compliant with FSR. It follows that the ISO 24610 standard is tied in with all standards related to linguistic content, be it morphosyntax, syntactic, or pragmatic content. Hence the standard contributes to the interoperability of the ISO standards for language resource management (Lee and Romary, 2010).

Acknowledgments

The writing of this paper has been supported by the Collaborative Research Center 991 'The Structure of Representations in Language, Cognition and Science' funded by the German Research Foundation (DFG).

References

Lou Burnard and Syd Bauman, editors. 2012. *TEI P5: Guidelines for Electronic Text Encoding and Interchange (Version 2.1.0)*. Text Encoding Initiative Consortium, Charlottesville, Virginia. [http://www.tei-c.org/Guidelines/P5/].

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

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row, New York.

Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Blackwell, Oxford.

ISO 24610-1. Language resource management - Feature structures - Part 1: Feature structure representation. Publication date: 2006-04.

ISO 24610-2. Language resource management - Feature structures - Part 1: Feature system declaration. Publication date: 2011-10.

ISO 24612. Language resource management - Linguistic annotation framework (LAF). Publication date: 2012-06.

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

Kiyong Lee and Laurent Romary. 2010. Towards interoperability of ISO standards for language resource management. In *Proceedings of the Second International Conference on Global Interoperability for Language Resources*, pages 95–103, Hong Kong. City University of Hong Kong.

Kiyong Lee, Lou Burnard, Laurent Romary, Eric de la Clergerie Thierry Declerck, Syd Bauman, Harry Bunt, Lionel Clément Tomaz Erjavec, Azim Roussanaly, and Claude Roux. 2004. Towards an international standard on feature structures representation. In *Proceedings of LREC 2004*, pages 373–376, Lisbon, Portugal. Universidade Nova de Lisboa.

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

Erik T. Ray. 2003. *Learning XML*. O'Reilly, Sebastopol, CA.

Eric van der Vlist. 2003. *RELAX NG*. O'Reilly, Sebastopol, CA.

Andreas Witt, Georg Rehm, Erhard Hinrichs, Timm Lehmberg, and Jens Stegmann. 2009. SusTEInability of linguistic resources through feature structures. *Literary and Linguistic Computing*, 24(3):363–372.