Proceedings of the Workshop on Resources in Tools and Field Linguistics, Third International Conference on Language Resources and Evaluation (LREC), May 26-27, Las Palmas, Canary Islands, Spain. Pages 25-1 to 25-10.

A Morphological Glossing Assistant

Mike Maxwell

Linguistic Data Consortium
3615 Market Street, Suite 200
Philadelphia, PA, 19104-2608, USA
maxwell@ldc.upenn.edu

Gary Simons

SIL International
7500 W Camp Wisdom Road
Dallas, TX 75236
gary_simons@sil.org

Larry Hayashi

Canada Institute of Linguistics/SIL International
7600 Glover Road,
Langley, BC V2Y 1Y1
larry_hayashi@sil.org

Abstract

One of the tasks language documenters face is that of assigning glosses to function morphemes, including affixes. These glosses are typically used in marking up interlinear text at a morpheme level. But without a morphological parser, marking up interlinear text is tedious and error-prone. Ideally, a parser will be guided not only by the form and syntagmatic properties of morphemes, but also by their morphosyntactic properties (features).

We describe a system which simultaneously helps the linguist use standard glosses for function morphemes, and assigns corresponding morphosyntactic features to those morphemes. These features can be used by a morphological (or syntactic) parser. Our system defines a mapping between glosses and features, as well as a way of extending the gloss/ feature system with properties which may have been overlooked. We illustrate the operation of the system from both the user's point of view and from an internal perspective.

1. Introduction

One of the tasks field linguists and other language documenters face is that of assigning glosses to function morphemes, including affixes. Among other applications, these glosses are typically used in marking up interlinear text at a morpheme level, as in the following example (taken from Morse and Maxwell 1999, page 44-5):

Waro-bo-RE 'bãrẽ
plant.sp-CLS:round-OBJ also

wo-Ij-Abẽ xoe-we
seek-STV-H/H.3M.Sg toucan-CLS:flat
'The toucan also looks for a certain (species of) plant.'

Marking up interlinear text by hand is tedious and prone to errors and inconsistencies. For that reason, interlinear text tools normally provide a morphological parser. But if this parser is guided only by the form and syntagmatic properties of morphemes, it may produce spurious parses. In English, for example, there are plausibly three affixes (or clitics) having the form *–s*, two having the form *–er*, etc.; usually only one of

these can plausibly be said to occur in a single word, but determining which one is correct can require morphosyntactic constraints.

Languages making more use of morphology than English tend to have even more ambiguity in parsing. If morphosyntactic constraints are ignored, spurious parses proliferate to the point where a parser becomes more cumbersome than helpful. It is therefore desirable to constrain the parser by the use of morphosyntactic properties (features).

However, language documenters (whether field linguists or native speakers) are often unfamiliar with linguistically motivated morphosyntactic feature systems. The result is a conflict: on the one hand, the parser needs a feature system; on the other, many documenters (particularly in the early stages of their work) do not want to have to build a possibly complex feature system, but would rather work with glosses.

In addition, language documenters would benefit from access to standards for encoding the meaning of functional morphemes, i.e. standard glosses. (Lexical morphemes—stems and roots—are glossed with general terms, for which it would not be feasible to provide standards.)

Theoretical linguists have developed linguistically based ontologies for such properties as case marking, gender systems, tense and aspect, etc. (Corbett 1991, Corbett 2000, Binnick 1991, Blake 2001, and many others). These ontologies can satisfy the need for standards for formal glossing.[1] We propose an additional role for ontologies, namely as the starting point for building a morphosyntactic feature system, thereby satisfying the need for a feature system to be used by the morphological parser (and in the future, by a syntactic parser). This dual use is made possible by the fact that there is—we claim—a fairly direct mapping between the ontology of morphosyntactic properties and a morphosyntactic feature system, and an even more direct mapping between morphosyntactic features and glosses.

Much of the work of specifying a universal terminology for morphosyntactic properties has been done, or is in progress in various projects, such as the E-MELD project (Lewis, Farrar and Langendoen 2001), and we intend to build on that foundation.

A mapping between a standard ontology and the features (or their corresponding glosses) will also facilitate comparison where glosses have divergent meanings in different traditions of linguistic description. For example, the term 'absolutive' has one meaning for linguists working on Nahuatl, and a different meaning for linguists describing ergative languages. Glosses can thus be defined by their mapping to a standard ontology of morphosyntactic properties.

We describe a system which assists the user in glossing function morphemes, using a standardized ontology of concepts. The system simultaneously provides a well-motivated but modifiable morphosyntactic feature system, usable by a sophisticated morphological (or syntactic) parser. Our system defines the mapping between glosses and features, and a mapping from these back to the ontology of morphosyntactic properties. The system also provides a way of extending the gloss/ feature system with properties not contained in the original ontology. While the system does not directly modify the original ontology on the basis of modifications to the gloss/ feature system, we do envision a human-mediated feedback system for possible extensions or modifications. (This feedback system is not, however, discussed in this paper.)

In addition to describing this mapping, we describe the user interface for glossing in section 3.

Our system is designed to be a component of a general knowledge base for describing languages called 'FieldWorks' (Hayashi and Hatton 2001) This knowledge base is based on many of the same underlying concepts as CELLAR (the Computing Environment for Linguistics, Literacy and Anthropological Research); see Simons (1998).

2. Overview of solution

A diagram giving an overview of our solution is shown in Figure 1. At the heart of the approach is an interactive tool called the Morphosyntactic Gloss Assistant. It takes two inputs: a general ontology of morphosyntactic concepts (which it displays in a 'Universal Gloss List Viewer'), and a language-specific feature system (which it displays in a 'Language Specific Gloss List Viewer'). Section 3 illustrates these two viewers and describes how the Morphosyntactic Gloss Assistant works. Section 4 goes behind the viewers to describe the conceptual model of the general ontology and the specific feature system.

When the user selects a particular gloss string from the language-specific gloss list, the system adds the corresponding feature to a feature structure. The full set of features for a particular morphosyntactic form constitute a complete feature structure. The mapping back from this feature structure to a gloss (which may consist of several individual gloss strings) for the morpheme is described in section 5.

[1] By 'formal' glosses, we mean glosses such as 'HAB' (for 'habitual') and 'DEF' (for definite), as opposed to such informal glosses as 'always' and 'the'. Cf. Simons and Versaw 1992 section 2.4.4.5 for this distinction

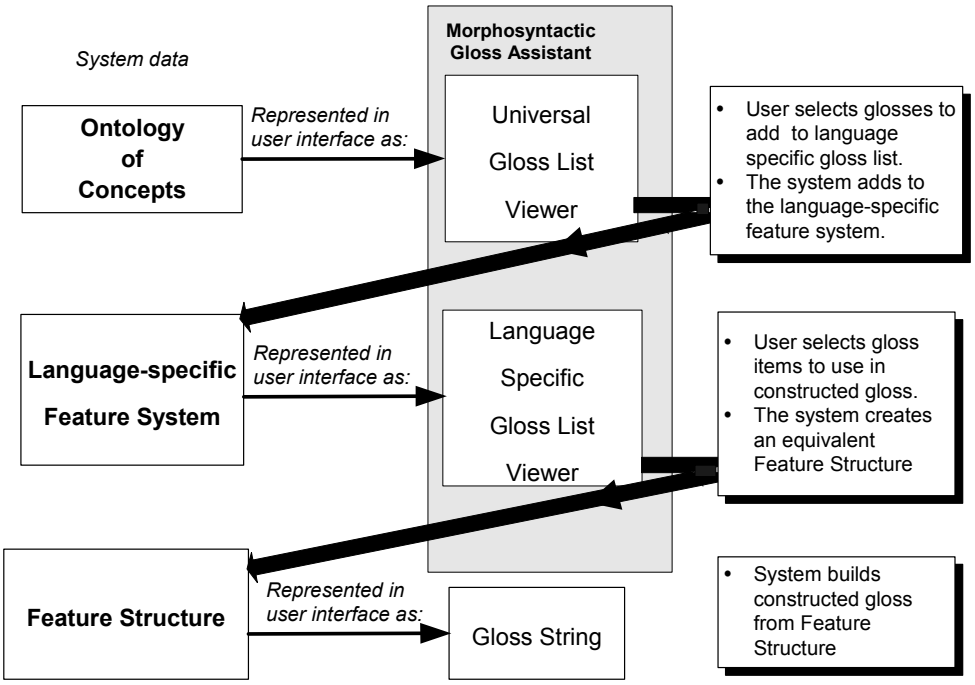

Figure 1. Overview of Morphological Glossing Assistant

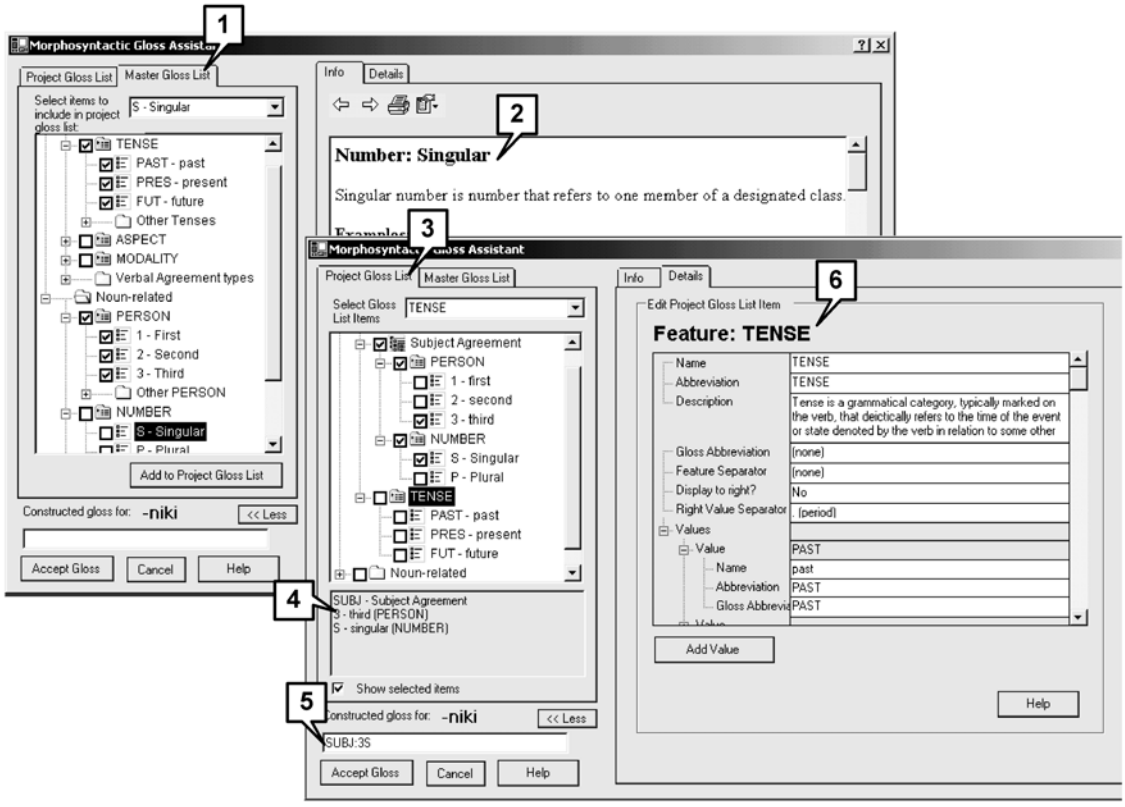

Figure 2: Morphosyntactic Gloss Assistant Interface

3. The user interface

Figure 2 presents the user interface of the Morphosyntactic Gloss Assistant. Features are noted via the numbered balloons and are explained below.

1. Clicking on the 'Master Gloss List' tab reveals a master ontology. Note that the most common values are displayed at the highest level. The user can find more less common values in the "Other <feature name>" folders.
2. The Master Gloss List includes documentation for each item in the list.
3. Once the user has selected items from the Master Gloss List, these items appear in the 'Project Gloss List' tab. The user selects items from here to build glosses for particular morphemes.
4. Glosses selected by the user are displayed below the Project Gloss List. When the user selects a gloss item, the system builds a corresponding feature structure (hidden here).
5. The system creates a gloss based on the feature structure it has built.
6. The user can define what the glosses and gloss separators are for individual features and their values.

The nature of the data models and the system operations that take place as the user carries out interface actions is described in sections 4 through 6.

4. Models of ontology, feature system and feature structures

Relating feature structures to gloss construction requires a data model that defines the following three inter-related subsystems:

1. The feature structures carried by the morphemes of the language;
2. The feature system of the language; and
3. An ontology of morphosyntactic concepts.

The user selects from the ontology, and the system creates the appropriate corresponding feature system objects. The user can customize these objects to reflect language-specific properties. Once feature structure types, features and their values have been added to the system, the user can then select from these to create the feature structures unique to each functor morpheme. The user can also add additional "gloss-specific" information to the feature system objects, and the system can then generate a of the feature structure as a gloss.

In this section we look at each of these data models in detail.

4.1. The model of feature structures

There are three types of morphemes that we want to help the user to gloss:

1. Inflected variants of stems (e.g. English men = 'man.PL');
2. Inflectional morphemes – particularly affixes and clitics (e.g. English –s = 'PL'); and
3. Derivational morphemes (e.g. English –ion = 'NMLZR' as an abbreviation for nominalizer).

The features and their specific values carried by each of these morpheme types are defined in a 'feature structure'. In the case of derivational morphemes, two feature structures are defined representing the before and after states of the derivation. For the purposes of this paper we will discuss glossing one the first two types of morphemes.

Figure 3 is a UML (Unified Modeling Language)[2] representation of the feature structure model. The classes, attributes and relationships are described below.

[2] Refer to http://www.holub.com/class/oo_design/uml.pdf for a summary of UML notations. In UML, object classes are represented as rectangles. Basic class attributes are displayed inside the rectangle. When a class is composed of other classes, the relationship is indicated by a line with a diamond on the composed class end. When one class refers to another class, the relationship is indicated with a simple line. A line with an arrow represents a subclass to superclass relationship. Abstract classes have their class name in italics.

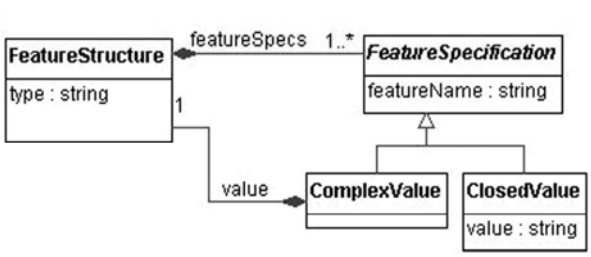

Figure 3. Data model for Feature Structures

FeatureStructure class

A FeatureStructure is a general purpose data structure which identifies and groups together the individual features of a given word or morpheme. FeatureStructures may be typed based on the kinds features the particular structure may take. The feature specifications of the FeatureStructure are specified by one or more FeatureSpecifications (see below).

FeatureSpecification abstract class

A FeatureSpecification associates a feature name with one or more values. Each FeatureSpecification can be one of two kinds, described below: a ComplexValue or a ClosedValue. Both kinds indicate the feature for which they specify a value.

ComplexValue subclass

A ComplexValue is a FeatureSpecification used for FeatureStructures that have nesting. It contains another FeatureStructure which is its value.

ClosedValue subclass

A ClosedValue is an instantiation of one of the FeatureValues (captured as the value association) for a particular ClosedFeature (captured as the feature association).

By way of illustration, consider with the feature structure in Figure 4, which might be associated with a verbal suffix:

$$\begin{bmatrix} \text{SUBJECT} \begin{bmatrix} \text{PERSON} : \text{third} \\ \text{NUMBER} : \text{singular} \end{bmatrix} \\ \text{OBJECT} \begin{bmatrix} \text{PERSON} : \text{first} \\ \text{NUMBER} : \text{plural} \end{bmatrix} \\ \text{TENSE} : \text{past} \end{bmatrix}$$

Figure 4: Standard feature structure notation for third person singular subject agreement and first person plural object agreement and past tense

We can represent this using a FeatureStructure, represented concretely by the XML fragment below[3].

```
<FeatureStructure type="TransitiveVerb">
 <featureSpecs>
  <ComplexValue feature="SUBJECT AGREEMENT">
   <value>
    <FeatureStructure type="Agreement">
     <featureSpecs>
      <ClosedFeature featureName="PERSON" value="third"/>
      <ClosedFeature featureName="NUMBER" value="singular"/>
     </featureSpecs>
    </FeatureStructure>
   </value>
  </ComplexValue>
  <ComplexValue feature="OBJECT AGREEMENT">
   <value>
    <FeatureStructure type="Agreement">
     <featureSpecs>
      <ClosedFeature featureName="PERSON" value="first"/>
      <ClosedFeature featureName="NUMBER" value="plural"/>
     </featureSpecs>
    </FeatureStructure>
   </value>
  </ComplexValue>
  <ClosedFeature featureName="TENSE" value="past">
  </ClosedFeature>
 </featureSpecs>
</FeatureStructure>
```

Figure 5. XML instantiation of feature structure

4.2. The model of feature system

The model thus far presented allows the linguist to create FeatureStructures without constraint: he can declare types, features and values as needed. Ideally, however, the building of FeatureStructures should be constrained by the grammar of the language. That is, there exists for each language a *feature system* that defines what types of feature structures are possible, what features those types capture, and what the possible values of those features are.

Figure 6 is a UML representation of our model for feature systems. The boxes in gray represent the FeatureStructure classes that we just described. Note, however, that what were simple string attributes in Figure 3 are now references to feature system objects in Figure 6: the FeatureSystem is thus used to constrain the FeatureStructures. The classes of the FeatureSystem are as follows:

FeatureSystem class

The FeatureSystem declares what FeatureStructureTypes exist in the language. For the purposes of morphosyntactic glossing (and parsing), we assume that only one FeatureSystem exists per language data project.

FeatureStructureType class

A FeatureStructureType is defined for each distinct type of FeatureStructure that exists in the data. FeatureStructureTypes are given a name, description and an optional abbreviation. In our glossing and parsing system, FeatureStructures specify the morphosyntactic features and values carried by function morphemes. Thus a FeatureStructureType declares what the possible features (FeatureDefn) are for FeatureStructures of that type.

[3] In this XML fragment, class names begin with capital letters. Basic attributes are show as attributes of the XML element, and relationships are shown as lower case XML elements.

FeatureDefn (abstract superclass)

Each feature of a FeatureStructureType can be one of several possible kinds. All FeatureDefns specify the a name (e.g. PERSON), a description, and an abbreviation (e.g. PERS). Below, we discuss the differences between the two most important subclasses.

ClosedFeature subclass

A ClosedFeature has a finite set of possible values. For example, in a particular language, the feature PERSON might specify the possible values of first, second and third.

ComplexFeature subclass

A ComplexFeature allows for nested feature structures. Languages often specify the same feature more than once on a single morpheme. For example, in Nahuatl, the verb indicates person and number agreement with both the subject and object (much like the FeatureStructure of Figure 4).

FeatureValue

FeatureValues are values of a particular ClosedFeature. They have a name, description, and abbreviation. For example, a language might specify singular, dual, trial and plural as possible FeatureValues for the NUMBER feature. For binary features, two FeatureValues are usually defined: a plus value and a negative value.

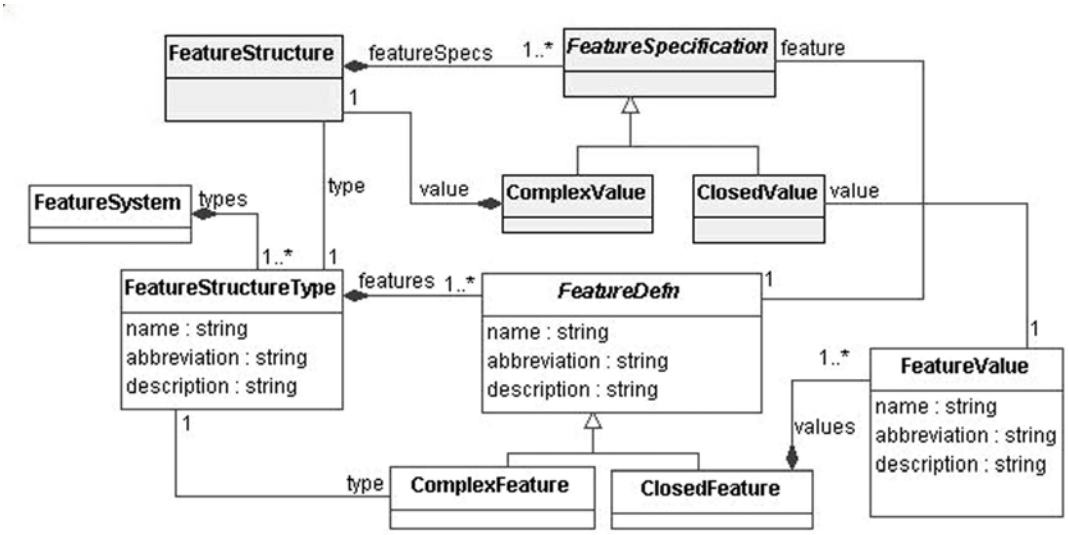

Figure 6: Feature System model

4.3. The model of ontology

The underlying model for the ontology is simple. It is essentially an outline of concepts, in which the embedding of one concept under another represents the "a kind of" relationship. Figure 7 shows a fragment of the ontology. For instance, working back from "close future", the ontology says that, "Close future is a kind of future, which is a kind of absolute tense, which is a kind of tense, which is a kind of verb-related property, which is a kind of morphosyntactic property."

As an object in the database, each concept has a number of attributes. An *abbreviation* proposes a standard abbreviation for use in glossing. A *definition* is available for display in the user interface. This supports the requirement that the glossing system help the user learn standard linguistic terminology. In addition, a unique *concept id* is given which is copied when creating the feature system. This allows for future cross-linguistic comparisons of feature systems against these ontological concepts, even if the language-specific feature system changes the name of the item.

```
morphosyntactic property
  noun-related property
    case
    definiteness
    noun class
    number
    person
    semantic role
  verb-related property
    aspect
    mood and modality
    polarity
    switch reference
    tense
      absolute tense
        future
          close future
          hodiernal future
          remote future
        past
        present
      absolute-relative tense
      relative tense
```

Figure 7. Ontology as outline.

5. From Ontology to Feature System

There is one more attribute on an ontological concept which is the key to automatically generating a feature system from an ontology. This is a *type* attribute; it specifies what the concept would correspond to in a feature system. The possible values for type are listed in Table 1.

Value	Meaning
fsType	The item corresponds to a feature structure type.
fGroup	The item is strictly for the purpose of grouping related features to ease navigation.
feature	The item corresponds to a feature.
complexN	The item corresponds to a complex feature that takes a feature structure of type “nominal”.
complexV	The item corresponds to a complex feature that takes a feature structure of type “verbal”.
vGroup	The item is strictly for the purpose of grouping related feature values to ease navigation.
value	The item corresponds to a feature value.
see	The item is not a possible gloss, but is a cross-reference to the gloss that should be used instead.

Table 1. Type attribute possibilities

In the Morphosyntactic Gloss Assistant, the ontology of concepts is displayed to the user in the Master Gloss List viewer (see section 3). When the user selects an item from the ontology to use as part of the current gloss, the type attribute instructs the MGA as to what it should do. For instance, referring back to

the ontology fragment in Figure 7, "absolute tense" is on type *vGroup* and selecting it should do nothing. On the other hand, "future" and the three more specific kinds of future below it are of type *value*; selecting one of these should add the selection as a possible feature value to the language-specific feature system. The MGA climbs up the ontology to find the concept of type *feature* that dominates the new value, in this case "tense", and adds it as a possible value of that feature. When a feature (rather than a value is selected) the resulting feature structure uses the built-in value *any*. Table 2 summarizes the way in which the eight concept types map onto the classes of the feature system.

Concept type	Can it be select-ed as a gloss?	Object class to generate in feature system
fsType	No	FeatureStructureType
fGroup, vGroup, see	No	None
feature	Yes	ClosedFeature
value	Yes	FeatureValue
coimplexN	Yes	ComplexFeature that takes a feature structure of type "nominal".
complexV	Yes	ComplexFeature that takes a feature structure of type "verbal".

Table 2. Mapping from types to feature system elements

The ontology is actually stored in an XML file. This makes it possible for the user to load in either a predefined global master list or a localized list that is more appropriate to the language family being studied.

6. From Feature Structure to Gloss

In order to represent a FeatureStructure as a gloss, we need to add a number of properties to the model. These additions are circled in Figure 8 and described below.

6.1. Additions to the FeatureStructure model

FeatureStructureType addition

Because we want gloss strings of a given morpheme to appear in a certain order (e.g. for a transitive verb, the gloss for subject agreement should precede the gloss for object agreement, and the gloss for person before that for number), the features that belong to a FeatureStructureType are ordered. Operations on the FeatureSpecifications within a FeatureStructure do not require any knowledge of order; thus this addition is purely for the sake of glossing.

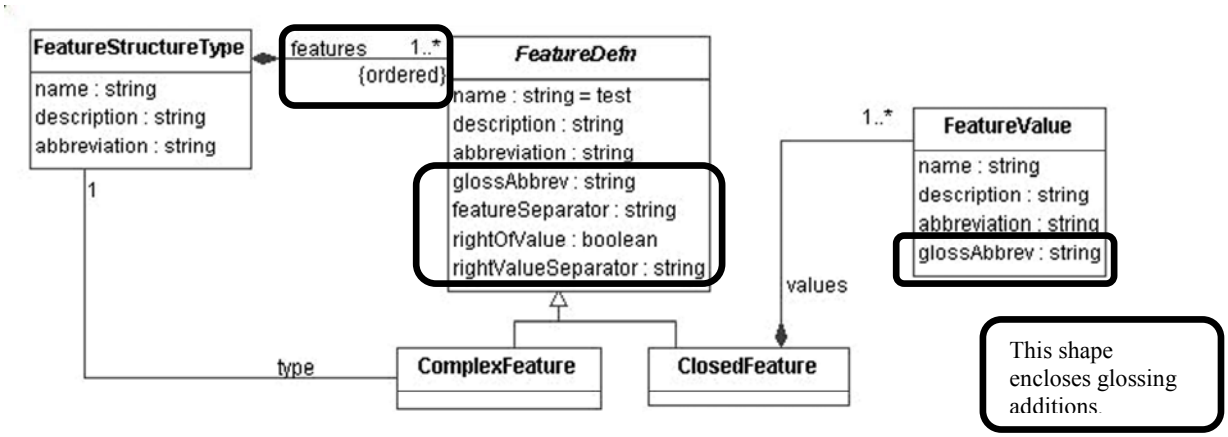

Figure 8. Additions to FeatureSystem model for glossing

FeatureDefn additions

For most cases, a gloss string will correspond to a feature value. But in some cases, it will be desirable for the gloss string to indicate the feature as well. In Tucanoan languages of Colombia, for example, most concrete nouns bear a shape classifier, of which there may be well over a hundred (Morse and Maxwell 1999). Many classifiers have the basic meaning of a lexical item, and it may not obvious whether the gloss of such a morpheme represents a lexeme or a classifier suffix. A solution is to use both the feature and its value as gloss: *papera-joka* 'paper-CLS:leaf", rather than 'paper-leaf". For these situations, a **glossAbbreviation** associated with a FeatureDefn may be defined.

When a feature glossAbbreviation does appear it can be optionally separated from its value with a **featureSeparator** (e.g. the colon in SUBJ:3S). The attribute **rightOfValue** allows the glossAbbreviation to occur to the right of the value, rather than to the left (e.g. 3S:SUBJ, +PLURAL).

The **rightValueSeparator** defines what occurs to the right of the gloss for a feature value if it is followed by another feature value. For example, the features of tense, aspect and modality are often found clustered in languages. One might want to separate each of these values with a separator (e.g. past tense, progressive aspect and irrealis modality represented as PAST.PROG.IRR). The rightValueSeparator on ComplexFeatures separates the entire "complex" from the next feature (e.g. the periods in SUBJ:3S.OBJ:1P.PAST).

FeatureValue addition

The FeatureValue itself also has a **glossAbbreviation**. The user can choose to not display certain features by leaving the attribute empty (e.g. a user might want to do this for default feature values such as present tense).

A situation where the use of an empty **glossAbbreviation** for a **FeatureValue**, and a non-empty **glossAbbreviation** for a **FeatureDefn**, occurs when it is desirable to indicate the general type of information some morpheme encodes, but not the details.[4] For example, in Spanish the full gloss of the verbal suffix *–o* might be '1Sg.PRES.IND' (for '1st Singular Present. Indicative'), but for some purposes the gloss 'Finite' may be adequate.

We could specify all the above behaviors on the objects of the FeatureStructure itself – that is for each morpheme that carries a feature structure, we would specify the abbreviations, separators, etc. for that specific gloss. However, one of the goals of our gloss assistant is to help the user be systematic in glossing. Thus, we specify the above behaviors on objects of the FeatureSystem rather than on the FeatureStructures.

6.2. Examples: model settings and resulting gloss views

Figure 9 demonstrates how the feature structure model is populated with data for the feature structure found in Figure 4. Note that the feature structure objects (in gray) contain no data but refer to feature system objects (indicated by the dotted lines[5]). The ComplexFeatures, ClosedFeatures and ClosedValues specify how the gloss is to be constructed, as described above. A period is used here to delimit the different gloss items except for the ClosedFeature of *person* (in accordance with the LSA stylesheet[6]).

6.3. Adding glosses absent from the ontology

While we expect to provide a wide range of glosses and corresponding morphosyntactic features, linguistics is not advanced enough to allow us to provide every feature necessary for all languages.

Shape classifiers (mentioned above) are a case in point: there are numerous languages with shape classifiers, and classifier systems are in principle open-ended. Thus, Cubeo (a language of Colombia) has separate classifiers for thread-like, rope-like, and vine-like objects; while Bora (a language of Peru) has classifiers for objects typically held with the teeth, and for things produced by cutting tools (Thiesen and Weber, in press).

Thus, we need to allow users to add glosses, along with the corresponding features. Our intention is for users to enter their glosses at a particular point in the hierarchy of glosses, to use the gloss as the feature value, and to use the super-node of that point in the hierarchy as the name of the feature.

Figure 2 illustrates the interface for this: the user clicks on the *Add Value* button and fills in the gloss information. For example, suppose the user decides to add a new tense gloss. In the figure, the user has

[4] Simons and Versaw 1992 (section 2.4.6.2) refer to such glosses as 'categories'.
[5] Note that this is not a standard UML diagramming convention.
[6] http://www.lsadc.org/language/langstyl.html

already specified one or more tense glosses; thus the feature for tense is already present in his language-specific list. After selecting this feature from the list, he can insert a new value with its gloss.

In the same way, the user can also add features to a feature type and specify both the gloss information for the feature itself and its values. For instance, suppose a hypothetical language inflects concrete nouns for the color of the object to which they refer. Since to our knowledge no language has been described with this sort of marking, we will not have included glosses for color marking. The user would need to create a node in the gloss hierarchy for 'color inflection'; this would become the name of the new feature. From there, the process proceeds as we have sketched above for the case where the feature-level node was already present in the hierarchy.

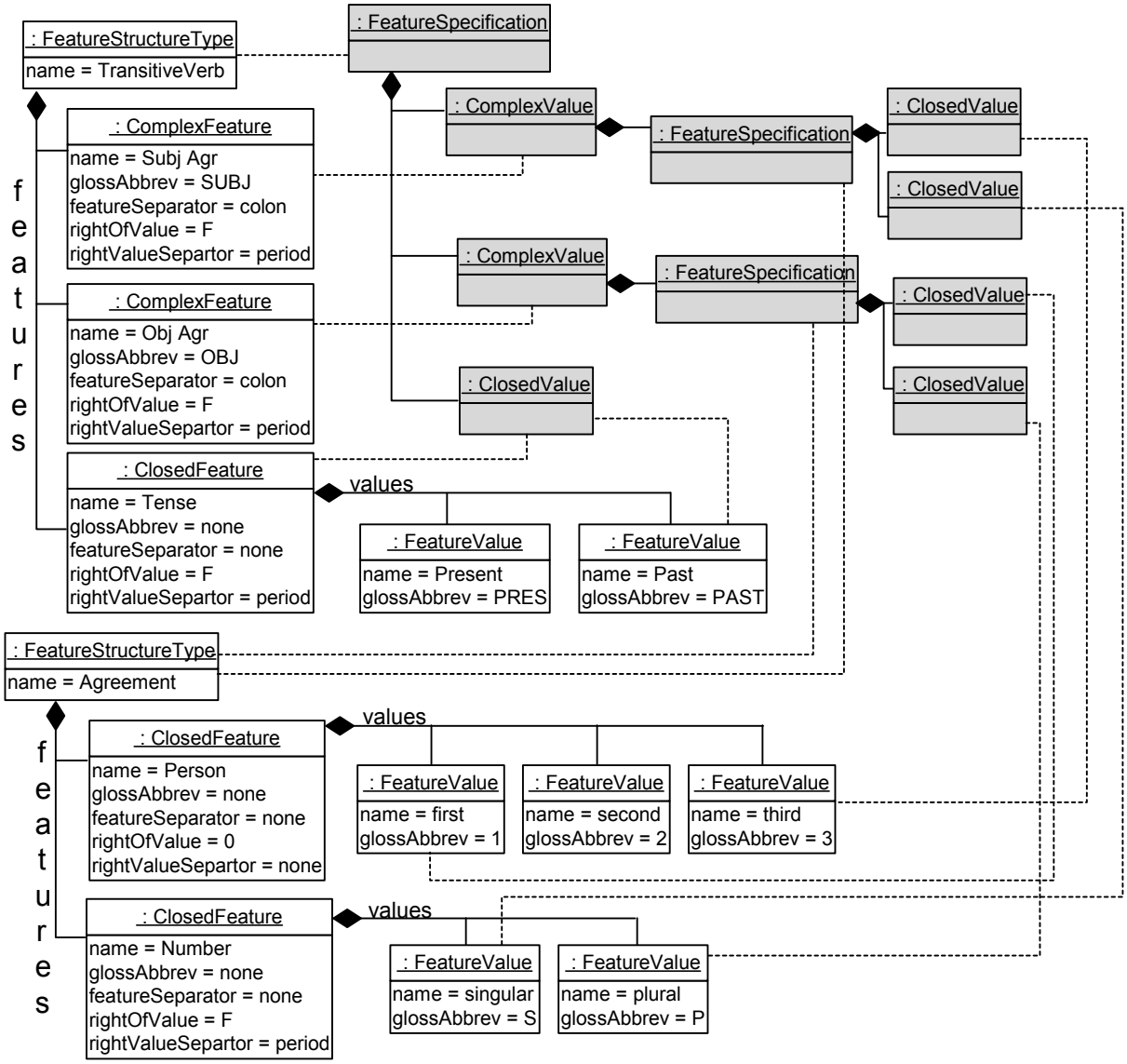

Figure 9 Instance diagram for SUBJ:3S.OBJ:1P.PAST

6.4. Limitations of the glossing model

While the glossing model is fairly flexible, there is at least one behavior which is not handled at present. This arises when there is an asymmetry between features and glosses.

In the most common case, a mapping between a feature value and a gloss string is one-to-one. For example, if a language distinguishes masculine and feminine genders, one plausible feature convention is to

have a binary feature GENDER, with the two values MASCULINE and FEMININE; another plausible convention is to have a binary feature FEMININE, with the two values + and –. In either case, there is a direct mapping from the two values to the glosses *masculine* and *feminine* (or their abbreviations).

The mapping between glosses and features may however fail to be one-to-one. For example, in languages which have both first person plural inclusive and exclusive forms, this distinction is usually indicated in the glossing by adding the word 'inclusive' or 'exclusive' (or some abbreviation) to the gloss for first person plural: '1PL.INCL' and '1PL.EXCL'. A direct translation of these glosses into morphosyntactic features would give something like:

$$\begin{bmatrix} \text{PERS} : 1 \\ \text{NUM} : \text{PL} \\ -\text{EXCL} \end{bmatrix}$$

and

$$\begin{bmatrix} \text{PERS} : 1 \\ \text{NUM} : \text{PL} \\ +\text{EXCL} \end{bmatrix}$$

where PERS(on) is a ternary feature having values 1, 2 and 3, and NUM(ber) is at least binary, having values SG and PL (and for some languages, dual, trial, paucal etc.). Under this feature system, it is necessary to introduce feature co-occurrence constraints to prevent the occurrence of one (or both) values of the EXCL(usive) feature with other than first person plural.

A different feature system has often been proposed (Matthews 1972, Anderson 1992, Noyer 1997) in which person is encoded by the binary features SPEAKER and HEARER. Under such a system, the features corresponding to the glosses '1PL.INCL' and '1PL.EXCL' would be the following:

$$\begin{bmatrix} +\text{SPEAKER} \\ +\text{HEARER} \\ -\text{SINGULAR} \end{bmatrix}$$

and

$$\begin{bmatrix} +\text{SPEAKER} \\ -\text{HEARER} \\ -\text{SINGULAR} \end{bmatrix}$$

respectively.[7] Thus, there is no direct map between the desired glosses and the morphosyntactic features.

We would need to add several capabilities to support the situation where there is an asymmetry between glosses and features. First, we would need to add rules to map between features and glosses. Fairly simple rules should be adequate, in which specific (extensionally specified) morphosyntactic feature-value sets map to a gloss string.[8]

Second, we cannot expect to provide a definitive morphosyntactic feature system which will satisfy everyone; advanced users must therefore be allowed to modify the morphosyntactic feature system, as well as the mapping between the glosses and those features.

7. Conclusion

We have described a tool, the Morpheme Gloss Assistant (MGA), which assists a language documenter to assign standardized glosses to function morphemes. While from the user's point of view, this is a glossing tool, at the same time the MGA builds a morphosyntactic feature system for the language and assigns morphosyntactic features to the glossed morphemes. These feature structures can be used by a morphological parser to eliminate spurious parses, thereby increasing the precision of parsing.

The MGA allows the user to choose glosses for a particular language from a linguistically motivated but language-independent ontology of morphosyntactic properties, and maps these to a language-specific

[7] The actual feature sets proposed by Matthews, Anderson and Noyer differ in various respects from each other and from the features shown in the diagram. Our focus here is not on the correct features, but on the asymmetry between glosses and features.

[8] There are open questions here, including how whole glosses would be parsed into separate gloss strings, and the possibility that the features corresponding to two glosses within a gloss string might conflict. We are aware of these issues, but do not address them in this paper.

feature system; it also allows additions by the user for language-specific properties. In addition, the MGA performs the mapping between language-specific feature values and language-specific glosses.

We have specified the design of the language-independent ontology and the language-specific feature system, as well as the mapping between these and the further mapping from feature structures to the language-specific glosses, using the Unified Modeling Language (UML). We have also sketched areas of our design which need further research. Since the system is still incomplete, we invite input.

8. References

Anderson, Stephen R. 1992. *A-Morphous Morphology*: Cambridge Studies in Linguistics, 63. Cambridge, Eng.: Cambridge University Press.

Binnick, Robert I. 1991. *Time and the verb: a guide to tense and aspect*. New York: Oxford University Press.

Blake, Barry J. 2001. *Case*: Cambridge textbooks in linguistics. Cambridge; New York: Cambridge University Press.

Corbett, Greville G. 1991. *Gender*. Cambridge England; New York: Cambridge University Press.

Corbett, Greville G. 2000. *Number*: Cambridge textbooks in linguistics. Cambridge, UK; New York: Cambridge University Press.

Hayashi, Larry S.; and John Hatton. 2001. "Combining UML, XML and relational database technologies - the best of all worlds for robust linguistic databases". In *Proceedings of the IRCS Workshop on Linguistic Databases*, eds. Steven Bird; Peter Buneman; and Mark Liberman, 115-124. Philadelphia: Institute for Research in Cognitive Science.

Lewis, William; Scott Farrar; and D. Terence Langendoen. 2001. "Building a Knowledge Base of Morphosyntactic Terminology". *IRCS Workshop on Linguistic Databases*, University of Pennsylvania, 150-156.

Matthews, P.H. 1972. "Huave Verb Morphology: Some Comments from a Non-Tagmemic Viewpoint". *International Journal of American Linguistics* 38:96-118.

Morse, Nancy L.; and Michael B. Maxwell. 1999. *Cubeo Grammar*. Studies in the Languages of Colombia, 5. Dallas, TX: Summer Institute of Linguistics.

Noyer, Rolf. 1997. *Features, Positions, and Affixes in Autonomous Morphological Structure*: Outstanding Dissertations in Linguistics. New York, NY: Garland. [MIT dissertation, 1992; distributed by MIT Working Papers in Linguistics].

Simons, Gary F. 1998. *The nature of linguistic data and the requirements of a computing environment for linguistic research.* In "Using Computers in Linguistics: a practical guide", John M. Lawler and Helen Aristar Dry (eds.). London and New York: Routledge, pp. 10-25.

Simons, Gary; and Larry Versaw. 1992. *How to Use IT: A Guide to Interlinear Text Processing*. Dallas: Summer Institute of Linguistics.

Thiesen, Wesley; and David J. Weber. In press. *A grammar of Bora*. Dallas: SIL International.