TREE UNIFICATION GRAMMAR

Fred Popowich
School of Computing Science
Simon Fraser University
Burnaby, B.C.
CANADA V5A 1S6

ABSTRACT

Tree Unification Grammar is a declarative unification-based linguistic framework. The basic grammar structures of this framework are partial descriptions of trees, and the framework requires only a single grammar rule to combine these partial descriptions. Using this framework, constraints associated with various linguistic phenomena (reflexivisation in particular) can be stated succinctly in the lexicon.

INTRODUCTION

There is a trend in unification-based grammar formalisms towards using a single grammar structure to contain the phonological, syntactic and semantic information associated with a linguistic expression. Adopting the terminology used by Pollard and Sag (1987), this grammar structure is called a *sign*. Grammar rules, guided by the syntactic information contained in signs, are used to derive signs associated with complex expressions from those of their constituent expressions. The relationship between the signs and the complex signs derived from grammar rule application can be expressed in *derivational structures*. These structures both explicitly illustrate relations that are implicit in the syntax of the signs and express relations that are present in the grammar rules.

Tree unification grammar (TUG) is a formalism which uses *function-argument (FA) specifications* as its primary grammar structures. These specifications resemble partially specified derivational structures of sign-based formalisms like head-driven phrase structure grammar (HPSG) (Pollard and Sag, 1987) and unification categorial grammar (UCG) (Zeevat, Klein and Calder, 1987). TUG uses FA specifications as lexical entries and possesses a single grammar rule which combines these specifications to obtain a specification for the complex expression being analysed. The use of FA specifications allows generalisations that are often captured in grammar rules to be captured in the lexicon.

MOTIVATION

The development of TUG was a consequence of investigating extensions to the UCG framework. As described by Zeevat, Klein and Calder (1987), UCG is a grammar formalism which combines some of the notions of categorial grammar with those of unification-based formalisms like HPSG and PATR-II (Shieber et.al., 1983).

The research reported in this paper was carried out at the University of Edinburgh under the support of a British Commonwealth Scholarship and at Simon Fraser University under an Advance Systems Institute Research Fellowship. Special thanks to the Centre for Systems Science and the Laboratory for Computer and Communications Research at Simon Fraser University for additional support. I would like to thank Dan Pass and the ACL reviewers for their comments and suggestions.

Like HPSG, the fundamental construction used in UCG is the *sign*. A UCG sign has attributes for *phonology*, *category*, *semantics* and *order*. Consider the sign for the expression *Mary walks* shown in (1).

(1) Mary-walks
sent[fin]
[e1] [[f1]mary(f1), [e1]walk(e1,f1)]
_

The *phonology* attribute of this sign (ie. *Mary-walks*) represents a phonological specification of the linguistic expression associated with the sign. For our needs we will use a simple sequence of words separated by hyphens. The *category* structure of a sign is very similar to that used by categorial grammar. There are three primitive categories, namely *sent*, *np*, and *noun*. Complex categories are of the form A / B, where B is a sign and A is a category (either primitive or complex). The *semantic* representation uses a language called InL (Zeevat, Klein and Calder 1987) which incorporates many of the features of discourse representation theory (Kamp 1981). An InL formula is of the form *[a]Condition* where *Condition* consists of a predicate name followed by its argument list. Each element of the argument list is either a variable (ie. discourse marker) or an InL formula. The variable *a* preceding *Condition* is the *index* of the formula. The *order* attribute of a sign contains information which is used to determine the ordering of the phonology of components during rule application. If an argument possesses *pre* as its order, then the phonology of the functor precedes that of the argument in that of the result. The value *post* describes the opposite situation. There is no restriction on the *order* of (1) as indicated by the appearance of the 'don't care' variable '_' in the order attribute.

InL variables are assigned *sorts*. A sort can be thought of as a collection of features based on factors like gender and number. Unification of variables of incompatible sorts will fail, thus providing a mechanism by which semantic information can restrict possible derivations. There are different sorts for *events*, *states* and *objects*. Variables of the object sort may be further specified with respect to gender (masculine, feminine, or neuter), and number. Unsorted variables will be denoted by the letter *a*, events by *e*, states by *s*, and genderless objects by *x*, *y*, and *z*. The letter *m* will be used to represent variables corresponding to a masculine object, *f* for feminine, and *n* for neuter. Unique identifiers which will be used to distinguish variables will appear as numbers following the variable names (ie. *n1*, *m1*, *s2*).

Signs may be underspecified and through the application of the grammar rules they may become increasingly specified by the merging of information. Only two grammar rules are proposed in (Zeevat, Klein and Calder, 1987):

(2) W_1-W_2: C: S: _ → W_1: C/(W_2:C_2:S_2:pre): S: _,
W_2:C_2:S_2:pre

(3) W_2-W_1: C: S: _ → W_2:C_2:S_2:post,
W_1: C/(W_2:C_2:S_2:post): S: _

They correspond to forward (2) and backward (3) functional application, the two rules in basic categorial grammar. Capital letters are used to denote *variables* that are associated with unspecified values which will be instantiated during a derivation. Colons are used to separate the different attributes of the sign when the sign is displayed in a horizontal rather than vertical manner. Consider the result of applying rule (3) to the two signs associated with *Mary* and *walks* which are shown below.

(4) Mary: np: mary(f1): _

(5) walks
sent[fin] / (_:np[nom]:[x]S:post)
[e1] [[x]S, walk(e1,x)]
_

The result of rule application is the sign that was introduced in (1). Rule application builds up the semantics of an expression by instantiating unspecified components, like *S* in the lexical entry for *walks* (5), that have been placed into the semantic structure.

Associated with every linguistic expression is a *derivation tree* which describes how the sign corresponding to the complete expression is derived from grammar rules operating over signs associated with lexical entries. The leaves of this binary tree are labelled with signs for individual words, the root is labelled by the sign for the complete expression, while the other nonterminal nodes are associated with intermediate expressions. Each nonterminal node is labelled with the result obtained by applying a grammar rule to the signs which are referred to by its two daughter nodes. The edges to the daughters of a nonterminal node are designated *functor* and *argument* depending on the role that the sign at the daughter node plays during grammar rule application.

As an example, the derivation tree provided in Figure 1 illustrates how backward functional application (BFA) (3) relates the signs for *Mary* (4) and *walks* (5) to the sign associated with *Mary-walks* (1). The functor edge of a nonterminal node is represented by a line darker than that of the argument edge. Rule application combines signs and builds derivation trees as a side effect. A more general form of this operation would be to combine trees to yield trees directly. *Partial descriptions* of a complete derivation tree could be combined to yield an increasingly further specified derivation tree.

The principle advantage of combining partial descriptions lies in the ease with which certain dependencies between different constituents can be described. Consider the general case in UCG where a functor is applied to an argument to produce a result. Each of these three constituents possesses its own set of features which describes the phonological, syntactic and semantic information associated with it (Bouma, Koenig and Uszkoreit, 1988). The relationship between these constituents is outlined in Figure 2. The information *F* associated with the functor can be dependent on the information *G* associated with the argument; the dependency relation is shown by the arc labelled ϕ in Figure 2. Such a dependency can be captured in the lexical entry for the functor since the functor contains the information associated with the argument in its own category name (as highlighted in bold in Figure 2). We have already seen an example of such a dependency in Figure 1 - the semantic information of the functor is dependent on that of the argument. While the dependency marked by ϕ can be captured in the lexicon in UCG, the dependency marked by ρ must be captured by the grammar rule; the grammar rule must state how the information *F'* associated with the result is obtained from that of the functor and that of the argument. If we adopt the premise that *F=F'*, then ρ becomes an identity relation and there is no need for introducing additional grammar rules to capture a more complicated relation ρ. Unfortunately, there are cases where the condition *F=F'* does not apply. For instance, Bouma (1988) argues for the need of a *lex* feature which would distinguish lexical elements from phrases; a lexical functor and its result would have different values for this feature (+*lex* and -*lex* respectively). Similarly, if one wanted to encode *bar level* information (Jackendoff, 1977) into the different constituents then there would be numerous cases where the bar level of a functor and that of its argument would not be the same. Most importantly though, we can provide a straightforward account of reflexivisation if we are not subject to the requirement that *F=F'* as we shall see shortly.

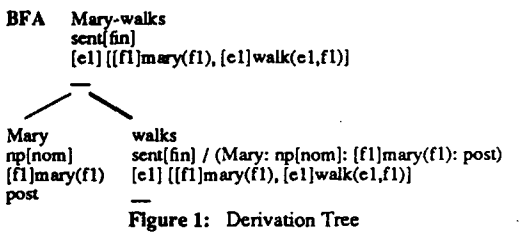

Figure 1: Derivation Tree

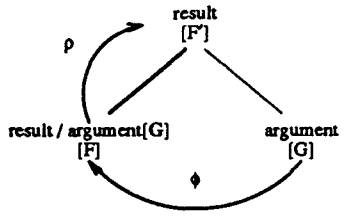

Figure 2: Dependencies Between Constituents

By using a partial description of a derivation tree as a lexical entry, dependencies corresponding to ρ in Figure 2 are captured in the lexicon instead of in the grammar rules. For instance, the BFA grammar rule states that the phonology of the resulting constituent consists of the phonology of the argument followed by that of the functor. The lexical entry for *walks* (5) implicitly describes such a relationship through the presence of the *post* feature. This feature is interpreted by the grammar rule, with the relation being explicitly represented in the result. If a partial description like the one introduced for *walks* in Figure 3 is used as a lexical entry, this relation is explicitly represented and the presence of a *post* feature is actually not necessary. Furthermore, local relationships other than those corresponding to ϕ and ρ can be captured explicitly in the lexical entry. For instance, the features associated with an argument can be dependent on those of its functor and information associated with the result can be directly related to that of the argument. One could even have a more long distance dependency, say between an argument and a subconstituent of its functor, stated directly in the lexical entry. Most importantly, the use of *FA specifications* similar to those introduced in Figure 3 allows us to capture the restrictions associated with reflexivisation in the lexicon, without requiring the introduction of additional grammar rules or principles.

FUNCTION ARGUMENT SPECIFICATIONS

Although the grammar rules operate over trees in TUG, signs still have a role to play in the organisation of information. The signs of TUG differ from those of UCG in several respects. First,

order information is not an explicit part of the TUG sign. The subcategorisation information that is contained in the UCG sign is not present in the TUG sign; it is represented in the tree structures of the framework instead. On a point of terminology, the second attribute of the TUG sign is referred to as the *syntax* instead of the *category*, since it contains more than just categorial information. Finally, the TUG sign will also contain an attribute for binding information. For now, however, we will restrict our discussion to only the first three attributes of a TUG sign.

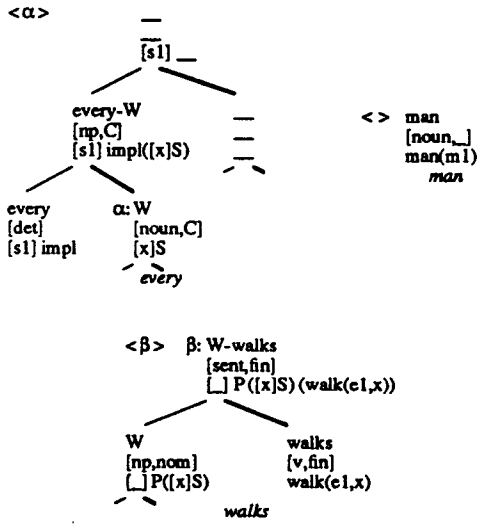

Figure 3: Lexical Entries

In TUG, a binary tree called an *FA specification* is associated with every linguistic expression. These specifications resemble partial descriptions of derivation trees. Each node of this binary tree is labelled with a sign. The root node possesses a sign corresponding to the complete expression, while the leaves are labelled with signs for the component words or morphemes. Each nonterminal node dominates a functor node and an argument node. The terms *functor-sign* and *argument-sign* will be used to refer to the signs associated with the functor and argument nodes respectively. The left-to-right ordering of functor and argument edges is not relevant! To refer to the sign of the root node of a tree, the term *root-sign* will be used. The trees rooted at nonterminal nodes of an FA specification will be called *subtrees*.

An FA specification contains an auxiliary list which specifies subtrees of the FA specification with which other FA specifications must be unified. It is represented as a list of labels contained in angle brackets appearing to the left of the FA specification as illustrated in the lexical entries introduced in Figure 3. Observe that there are two edges leading from the functor-sign of the FA specification for *every* which do not lead to any nodes. These *hanging edges* are associated with nodes whose *terminal* or *nonterminal* status has not yet been established. So an FA specification may either state that a constituent has no subconstituents (terminal node sign), it may state that it has subconstituents (nonterminal node sign), or it may say nothing about whether or not a constituent possesses subconstituents (node with hanging edges).

The single grammar rule of TUG is introduced in (6), where H_α denotes an FA specification with auxiliary list α.

(6) $H_\alpha \rightarrow H_{[C|\alpha]}, C_{[\,]}$

It describes how the FA specification for a complex linguistic expression is obtained from unification of the FA specifications associated with component expressions. This rule states that an FA specification *C* (which will be called the auxiliary tree) possessing an empty auxiliary list *[ ]* is unified with the subtree of *H* described by the first element of the auxiliary list of *H*. *[C/α]* denotes the list formed by adding *C* to the front of the list α. The result of this rule is a more fully instantiated version of the *primary tree*, *H*. The result's auxiliary list will consist of all but the first element of the auxiliary list of the primary tree. Viewed procedurally, this rule states how to construct a new FA specification from two pre-existing FA specifications. Declaratively, the rule merely states a relationship between FA specifications. To illustrate how FA specifications are manipulated by this single grammar rule we will trace the construction of the FA specification associated with the sentence *Every man walks*, using the lexical entries introduced in Figure 3.

The lexical entry for *every* requires an auxiliary tree to be unified at the location marked by α. For the moment, let us examine the subtree associated with the argument of the lexical entry. This subtree describes a functor-argument relation between two linguistic expressions. One is a functor *noun* of unspecified case *C* possessing an index compatible with the 'entity' sort, as designated by the presence of *x*, while the other is an argument determiner with phonology *every*. Alternatively, one could view the determiner as a functor over the noun as suggested in (Popowich, 1988). However, treating the noun as the functor allows a uniform treatment of nouns with possessive determiners and those with 'regular' determiners. This is the same treatment that has been adopted in HPSG (Pollard and Sag, 1987). We will propose that for any subtree the functor-sign and the root-sign will generally possess the same syntactic category information, except for *bar-level* information (Popowich, 1988), in a manner reminiscent of the *head feature convention* of GPSG (Gazdar et.al., 1985). Observe that the phonology of the root-sign of this subtree is that of the argument-sign followed by that of the functor-sign. The argument-sign introduces a semantic index of the 'state' sort which will also be the index of the InL formula of any constituent which possesses a universally quantified noun phrase as its argument. This means that sentences like *Every man walks* will describe a state, even though the word *walks* describes an event. This argument-sign also introduces the semantic connective *impl* which is associated with the universal quantifier.

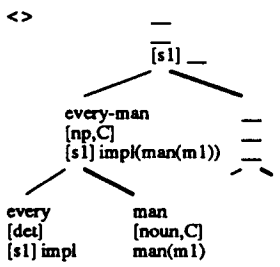

Figure 4: Intermediate FA Specification

When the FA specification for *man* is treated as a (depth zero) auxiliary tree which is unified with α from the lexical entry for *every*, we get a more instantiated FA specification which is associated with *every man*. This specification, which is introduced in Figure 4 is similar to the lexical entry for *every* except that *x* has been instantiated to *m1*, *S* to *man(m1)*, and *W* to

man. It also differs from the lexical entry for *every* in that it does not possess any labelled subtrees with which an auxiliary tree could be unified. As an abbreviatory convention, the index preceding a predicate which contains the index as its first argument will be omitted. So *man(m1)* is actually an abbreviation for *[m1]man(m1)* and *walk(e1,x)* is an abbreviation for *[e1]walk(e1,x)*.

The FA specification for *every man* can act as an auxiliary tree to be unified with β from the lexical entry for *walks* shown in Figure 3. Any potential auxiliary tree must have an argument-sign whose syntax is compatible with the 'nominative noun phrase' specification. No restrictions are placed on the indices of the root and argument signs; these indices will be specified by the auxiliary tree. The lexical entry for *walks* states how the semantics of the root-sign is formed from that of its functor and argument signs. When the FA specification for *every man* is combined with this primary tree, *P* of the primary tree is unified with *impl* of the auxiliary tree, *x* is instantiated to *m1*, and *S* is unified with *man(m1)*. *C* of the auxiliary tree is instantiated to *nom*. The resulting FA specification is shown in Figure 5.

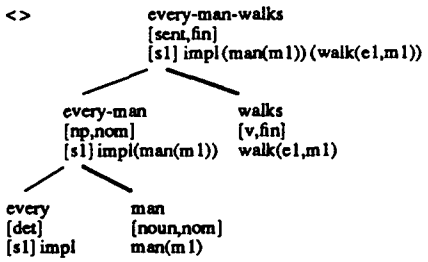

Figure 5: Final FA Specification

The FA specification for the complete sentence describes exactly one *FA structure*. While FA specifications may contain variables and partially instantiated attributes, FA structures do not. The lexical entries of TUG can be viewed as contributing constraints to the FA structure that is associated with a complex linguistic expression with the single grammar rule being used to combine these constraints. During the analysis of an expression, constraints are continually proposed and never rescinded. Eventually, these constraints will describe the final FA structure(s). Thus we distinguish between *information structures* and the *descriptions* of those structures in a manner similar to the approach proposed by Kaplan and Bresnan (1982) and discussed in detail by Johnson (1987). An FA specification can be interpreted as describing a set of FA structures. Grammar rule application then corresponds to the intersection of the sets associated with the component FA specifications. The resulting set is associated with a new FA specification. If the resulting set contains no FA structures, then there is no FA specification associated with the resulting set - grammar rule application fails! An ungrammatical sentence (ie. one without an FA structure) will not be assigned an FA specification. The result of the grammatical analysis of a sentence is the set of FA structures described by the final FA specification. Grammatical sentences can have one or more FA specifications, each of which will describe at least one FA structure.

We are requiring a wellformed FA specification to describe at least one FA structure. In this respect, FA specifications differ from the description languages introduced in (Kasper and Rounds, 1986) and in (Johnson, 1987). These languages allow descriptions for which there may not be associated structures. FA specifications are actually higher order descriptions which may be defined in terms of these description languages. They are intended to (transparently) describe structures associated with linguistic expressions; they are not intended to be a powerful language for describing feature structures in general. Instead of using FA specifications to describe FA structures, we could use one of these lower level description languages in conjunction with a restriction requiring a wellformed description to describe at least one structure.

In TUG, many local dependencies between grammatical constituents and some other bounded relationships can be stipulated explicitly in lexical entries. This is because FA specifications for one lexical entry can directly access information contained in the sign associated with a different linguistic expression. For instance, we have already seen how the lexical entry for a quantifier can directly specify semantic information (the index) for a sentence in which it is contained. It is possible to incorporate the constraints on reflexivisation perspicuously in the lexicon without causing unnecessarily complicated lexical entries and without requiring the introduction of additional principles or grammar rules.

REFLEXIVE ANTECEDENT INFORMATION

The TUG treatment of reflexives will be based on the concept of reflexive antecedent information, henceforth *R-antecedent information*. R-antecedent information, which will be distinct from the semantic information contained in a sign, will be responsible for determining the antecedents of reflexive pronouns. The constraints on reflexivisation will determine how the R-antecedent information of one sign is related to the information contained in other signs of an FA structure.

Since the signs corresponding to the reflexive and its antecedent need not both be present in the FA specification for a verb (as illustrated in sentences like *John wrote a book about a picture of himself*), we will introduce a *reflexive attribute* into the TUG sign. This 'binding' attribute will contain the R-antecedent information needed for establishing an anaphoric relationship between the reflexive and its antecedent. Since we have already seen the type of information contained in the first three attributes of the sign, let us consider the information contained in the fourth attribute.

The antecedent information is responsible for determining the discourse marker that can be the antecedent of the pronoun. Based on a proposal for the treatment of personal pronouns described in (Johnson and Klein, 1986) we will propose that the R-antecedent information explicitly describes the set of potential discourse markers available as antecedents for reflexives. This is the information that will be contained in the reflexive attribute of a sign. The lexical entry for the reflexive will only need to state that its antecedent marker is an element from this store. Unlike the *Cooper storage mechanism* described in (Cooper, 1983) which has been adopted in various proposals for anaphora (Bach and Partee, 1980, Gazdar et.al., 1985), our reflexive attribute contains a set of antecedents, not a set of anaphors.

The R-antecedent information will be represented as an ordered list of discourse markers (sorted variables) corresponding to potential antecedents. Lists will be displayed in square brackets with the different elements separated by commas. The notation *[...x/_]* will be used to designate *x* as an arbitrary element from a list with *[x/A]* denoting the list resulting from the addition of an element *x* to a list *A*. The sign associated with a reflexive

pronoun will resemble the one shown in (7).

(7) himself
[np, obj]
true(m)
[... m | _]

The discourse marker appearing in the semantic formula associated with the reflexive pronoun is an arbitrary element (of the masculine sort) of the reflexive attribute of the pronoun. The condition *true* introduced in the semantic attribute is always satisfiable for any discourse marker. We will discuss the semantics of the reflexive pronoun in more detail shortly.

The operation of selecting an arbitrary element from a list of arbitrary length is a fairly powerful operation. Nevertheless, it seems to be a sufficiently primitive operation to be included in a framework. It cannot be expressed in the PATR-II framework (Shieber et.al., 1983) which is often used to implement grammars. If functional uncertainty (Kaplan, Maxwell and Zaenen, 1987) were included as a primitive in PATR-II, then this arbitrary element selection operation could be implemented.

The constraints on reflexivisation, which affect the distribution of R-antecedent information and its interaction with other forms of information, are incorporated directly into the TUG lexical entries. One constraint is derived from Keenan's (1974) proposal whereby the antecedent for a pronoun is an argument of the functor containing the pronoun. This can be incorporated into TUG by having the R-antecedent information of a functor consist of the R-antecedent information of its parent sign augmented with the semantic index[1] of its argument. To illustrate this 'flow' of R-antecedent information, consider an analysis of the simple sentence *Mary loves herself.*

A series of FA specifications corresponding to different stages of an analysis for this sentence are shown in Figure 6. To highlight the relevant information, much of the information contained in the signs of these FA specifications has not been displayed. The first FA specification corresponds to the lexical entry for *loves*. Observe that the R-antecedent information of the functor-sign consists of the semantic index of the argument sign; the reflexive attribute of the sign associated with the object noun phrase is the same as that of the constituent which contains it.

[1]The detailed account of reflexivisation described in (Popowich, 1988) uses the *anaphoric index* instead of the *semantic index* of the argument. Since these two indices are identical in most cases, we will simplify our discussion by using the semantic index.

Also note that the InL formula from the sign associated with the verb references the semantic indices of the signs for the two noun phases. The second FA specification from Figure 6 illustrates the effect of unifying a sign (actually a depth zero tree) corresponding to the noun phrase *Mary* with the argument-sign of the initial FA specification. Note that the semantic index, *f1*, of *Mary* is introduced into the reflexive attribute of the functor over *Mary*. It also appears as the second argument of the semantic predicate *love* (underlined in the FA specification). Since the lexical entry for the verb also embodies the relation requiring the reflexive attribute of an argument-sign to contain the same information as its parent sign, *f1* is also introduced into the sign associated with the object noun phrase. This 'flow' of R-antecedent information is highlighted by the dark arrows in Figure 6. In the final FA specification from this figure, a sign corresponding to the reflexive pronoun is unified with the sign of the object noun phrase in the FA specification. The reflexive pronoun obtains its semantic index from the information contained in its reflexive attribute as highlighted by the small arrow. This semantic index is used as the final argument in the InL formula associated with the verb (which is underlined in the FA specification).

By incorporating Keenan's (1974) proposed dependency into FA specifications in this manner, we obtain a relationship much like *predication-command* (Hellan, 1988) and *F-command* (Chierchia, 1988). Although these 'command' restrictions on reflexivisation can account for much of the data concerning the distribution of reflexive pronouns, additional restrictions are necessary (Popowich, 1988). Just as the syntactic c-command relation needs to be used in conjunction with a locality restriction (eg. the syntactic 'clause-mate' restriction), the distribution of R-antecedent is restricted by a *semantic* locality restriction. Such a restriction, which is proposed in Pollard and Sag (1983), essentially states that reflexive 'information' cannot pass through categories of a *generalised predicative* type. A generalised predicative takes an NP denotation as its argument, and returns either an NP denotation or a 'proposition.' Adopting the notation used in (Dowty, Wall and Peters, 1981), the semantic type of a functor that takes expressions of semantic type α as arguments to produce resulting expressions of type β is $<\alpha,\beta>$. This means that the semantic type of a generalised predicative is either $<NP',NP'>$ or $<NP',S'>$, where NP' and S' are the semantic types associated with noun phrases and sentences respectively. Conventional categories that are associated with generalised predicatives include possessed nominals (like *picture of himself* in the phrase *John's picture of himself*) and verb phrases.

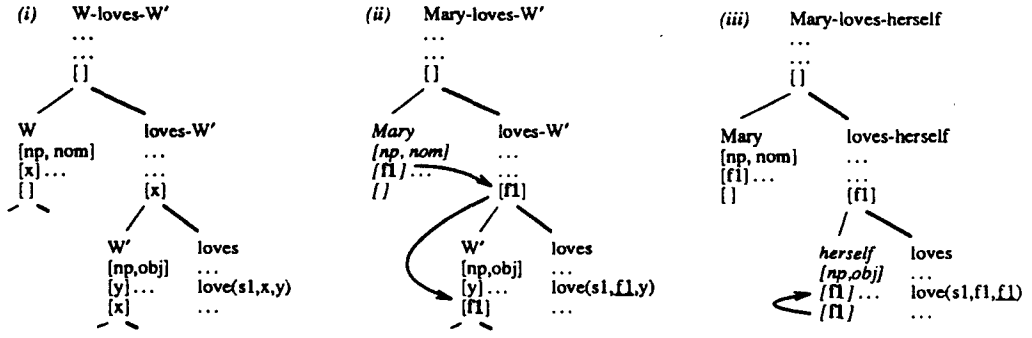

Figure 6: Distribution of R-Antecedent Information

The presence of a generalised predicative results in the blocking of R-antecedent information. Consider a subtree of an FA specification (like α in Figure 7) where the functor-sign is a

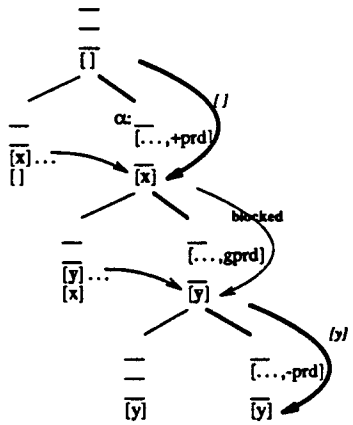

Figure 7: Predicate-Command and Locality Restrictions

generalised predicative. The R-antecedent information of the generalised predicative is a list consisting of only the semantic index of the argument-sign. The R-antecedent information of the root-sign does not contribute to that of the functor sign. The signs of an FA specification corresponding to generalised predicative functors will be marked with a syntactic feature to distinguish them from non-generalised predicatives. Functor-signs will be marked with the feature *gprd* if they are generalised predicatives. Non-generalised predicative functors which take noun phrases as arguments will be marked as *+prd*, and other functors will possess the feature *-prd*. Arguments will not be marked with any 'predicate' features. These features are not actually necessary for our account of the distribution of reflexive pronouns; our restrictions on reflexivisation can be defined in terms of other basic features. The use of these features will allow the behaviour of R-antecedent information to be observed more easily, as illustrated in Figure 7.[2] For predicative functors, the R-antecedent information of the functor-sign is composed of the semantic index of the argument-sign and the R-antecedent information from the root-sign. Note that the R-antecedent information of the sign labelled α is not included in that of the generalised predicative, but the semantic index of the argument-sign of α is included in that of the functor. For non-predicative functors, the R-antecedent information of the root-sign will be the same as that of the functor-sign.

AN EXAMPLE

Now that we have seen how R-antecedent information can be incorporated into FA specifications, we can examine how this information interacts with other forms of information during the analysis of a more complex sentence. We shall consider the analysis of the sentence *Mary loves a picture of herself*. After introducing various lexical entries, we shall see how they are combined with lexical entries introduced earlier in this paper to form more complex FA specifications.

[2]Instead of incorporating these three different relationships directly in the various lexical entries, they can be embodied in *lexical templates* which can be used in lexical entries (Shieber et.al., 1983, Popowich, 1988). All of the lexical entries introduced in this paper can be simplified through the use of templates.

In the lexical entry for *herself* in Figure 8, it is the argument-sign that is associated with the linguistic expression *herself*. This sign contains a restriction *[...f/_]* which specifies that the semantic index *f* associated with *herself* is a member of the reflexive attribute of the sign. This arbitrary element of the reflexive store is required to be a variable of the feminine sort. The syntax of this sign states that *herself* can act only as a noun phrase of the objective case. Thus it cannot appear in any positions in an FA specification which require the noun phrase to possess some other case, like *nominative*. Like other noun phrases, the argument-sign contains the semantic connective *and* which will be used in determining the semantics of the root-sign. Unlike lexical entries for proper names and quantified noun phrases, the semantics of the argument-sign does not associate any restrictive condition on the index it introduces; the condition *true* is always satisfiable for any discourse marker. This ties in with the view of pronouns being semantically underspecified linguistic items. Viewed in terms of DRT (Kamp, 1981), the formula *true(f)* (which is an abbreviation for *[f]true(f)*) merely introduces a discourse marker into the *universe* but does not introduce any *condition* on that marker. Since the syntax of our semantic notation requires a formula to consist of an index-condition pair, we need to introduce a condition like *true* along with the discourse marker.

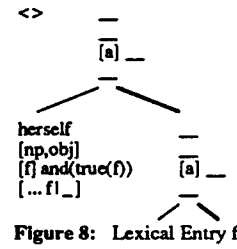

Figure 8: Lexical Entry for *herself*

The lexical entry for the 'depictive' preposition *of*, which is used in picture-noun constructions, is introduced in Figure 9. *Of* takes an object noun phrase argument to form a constituent which modifies a common noun. Additional restrictions would be required to ensure that it modifies only depictive nouns like *picture* and *portrait*. The lexical entry requires an auxiliary tree corresponding to an object noun phrase to be unified with α and one for a noun to be unified with β. It also introduces a semantic formula *of(x,y)* which requires the entity denoted by *x* to be *of* the entity denoted by *y*. Semantic formulae of the form *[a][A,B]* are abbreviations for formulae of the form *[a]and(A)(B)*. The functor-sign of α has been specified as a generalised predicative - it takes a noun phrase as an argument and results in another noun phrase. According to our restrictions on R-antecedent information, the R-antecedent information *A* of the root-sign of α is not included in that of the generalised predicative but it is included in that of the argument-sign. In this way, the same R-antecedent information that is associated with the root-sign of α is also available to the embedded noun phrase (ie. the argument of α) as highlighted in bold in Figure 9. The functor-sign of the lexical entry for *of* possesses the feature *+prd* since it takes a noun phrase as its argument to produce a noun. Since an argument sign always inherits its R-antecedent information from the root-sign, the same R-antecedent information is associated with both the root-sign of the lexical entry and the embedded noun phrase.

In order to obtain the FA specification for *picture of herself* shown in Figure 10, the lexical entry for *herself* acts as the

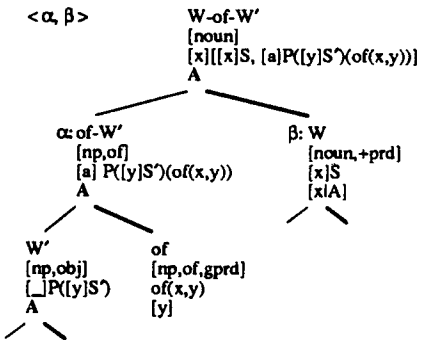

Figure 9: Lexical Entry for *of*

auxiliary tree which is unified with α of the lexical entry for *of*, and the lexical entry for *picture* is unified with β. Since *[f]and(true(f))* is an abbreviation for *[f]and([f]true(f))* in Figure 8, the unification of this formula with *[_]P([y]S')* from the primary tree will result in *P* becoming instantiated to *and*, *y* to *f*, and *S'* to *true(f)*. Note that in this example, *P* is a variable over our (finite) set of semantic connectives. The FA specification for *herself* introduces a restriction on the reflexive attribute of the sign associated with *herself*. This restriction requires *f* to be a member of the list *A* which is still uninstantiated. To represent that the restriction *[... f | _]* was unified with *A*, we will introduce *A* as a subscript on this restriction in the FA specifications that we are discussing. This will make it easier to examine the behaviour of R-antecedent information. The lexical entry for the noun *picture* introduces a marker of the neuter sort, *n1*, and includes a condition which requires this marker to be a picture *pic(n1)*. When this lexical entry is combined with the FA specification for *of herself*, *x* from the primary tree gets instantiated to the variable associated with the picture *n1*. Note that *[n1]and(true(f))(of(n1,f))* is equivalent to *[n1]of(n1,f)*.

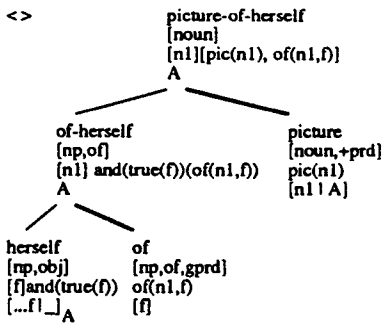

Figure 10: FA Specification for a picture-noun

The FA specification for the determiner *a* is very similar to the one for the universal quantifier introduced in Figure 3. We will not discuss it in detail here. Instead we will just note that it is constructed so that the reflexive attribute of the root-sign of the FA specification for the phrase *a picture of herself* will be the same as that of the sign associated with the complex noun *picture of herself*. Since the reflexive attribute of the sign associated with this complex noun is the same as that of the embedded reflexive noun phrase (see Figure 10), this means that the R-antecedent information, *A*, of the complex noun phrase *a picture of herself* is the same as that of the embedded noun phrase associated with the reflexive pronoun. So, any antecedents available to the complex noun phrase will also be available to the embedded reflexive. This will result in the appropriate distribution of R-antecedent when the FA specification associated with *a picture of herself* acts as an auxiliary tree to be combined with the primary tree corresponding to the lexical entry for *loves*.

The lexical entry for the transitive verb *loves* (Figure 11) requires two auxiliary trees corresponding to its object and subject noun phrases to be unified with subtrees α and β respectively. It is structured in much the same way as the lexical entry for *walks* discussed earlier. Note that for α, the functor-sign is not a generalised predicative and so the R-antecedent information of the functor sign is made up of the semantic index *y* of the argument-sign and the R-antecedent information *[x]* of the root-sign. β does have a generalised predicative functor-sign, so the R-antecedent information *A'* of the root sign is not included in that of the generalised predicative, *[x]*.

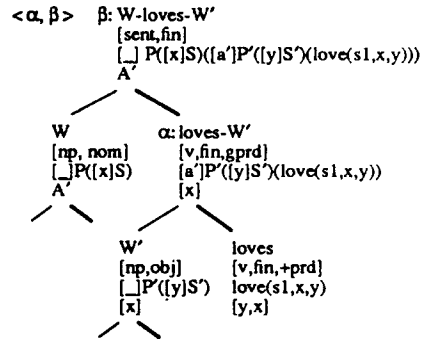

Figure 11: Lexical Entry for *loves*

When the lexical entry for *loves* takes the FA specification for *a picture of herself* as an auxiliary tree to be unified with α, the reflexive attribute *A* from the auxiliary tree becomes instantiated to *[x]*. But recall that there is still an additional restriction placed on the *A* which requires *f* to be an arbitrary member of *A*. This means that *f* must be unified with *x*; the subject of the verb is stipulated to be an entity possessing a marker of the feminine sort as illustrated in Figure 12. Unification of the auxiliary tree with α also results in *y* being instantiated to the variable associated with the picture *n1*. The semantic formula *PIC(n1,f)* in Figure 12 is an abbreviation for the somewhat lengthy formula *[n1] [pic(n1), of(n1,f)]*.

When the FA specification from Figure 12 is combined with the auxiliary tree corresponding to the lexical entry for *Mary*, the variable *f* from the primary tree becomes instantiated to the discourse marker associated with *Mary*. An attempt to unify an FA specification for a 'masculine' noun phrase with β of the primary tree would fail since the nominative noun phrase is required to possess a semantic index of the feminine sort (as shown in bold). Thus, for a sentence like *John loves a picture of herself* there would be no FA specification and consequently no FA structure (unless there were some female entity named *John*).

COMPARISON

The name "Tree Unification Grammar" suggests that TUG might be related to other unification-based frameworks as well as to other tree-based frameworks. We shall briefly compare TUG with some of the better known of these related frameworks. A

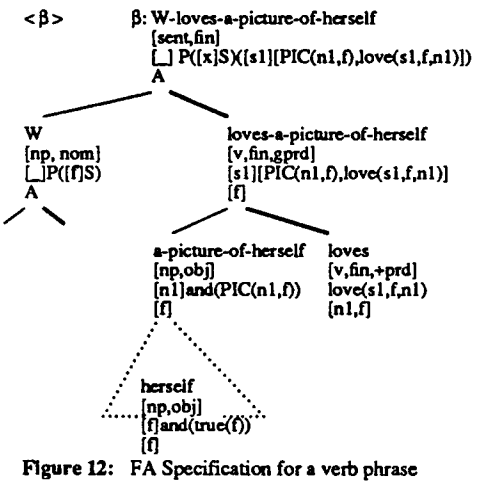

Figure 12: FA Specification for a verb phrase

more detailed discussion can be found in (Popowich, 1988).

Uszkoreit (1986) introduces Categorial Unification Grammar (CUG) as a class of grammars which combine the features of categorial grammars with those of unification grammars. In CUG, directed acyclic graphs (DAGs) are used as the basic grammar structures. Grammatical constituents possess attributes for *phonology*, *syntax*, and *semantics*. These constituents are essentially the *signs* of CUG. Two grammar rules, for forward and backward functional application, are used to form new constituents. CUG is similar to PATR-II in that it could serve as a language into which TUGs could be translated. A potential disadvantage of CUG is that it might be too unrestricted in the type of operations that it allows (van Benthem, 1987). In addition, the type of structures allowed in TUG is very restricted (binary trees containing only a fixed number of attributes) while those allowed in CUG are much less restricted. The structures used by TUG, UCG and other formalisms can be translated into a low-level format consisting of CUG DAGs. A major shortcoming of using CUG or PATR-II as a linguistic formalism is that the dependencies that are necessary for determining anaphoric relationships are 'hidden' in the DAG describing the linguistic expression; information is distributed in a flat graph structure with no higher order grouping expressed. Although this may be beneficial with respect to implementing grammars, it can make it difficult to work with the structures. The advantage of the FA structure is that it is an explicitly hierarchical representation structure - a tree with structured nodes - instead of a graph of simple nodes. This hierarchical structure allows many linguistic generalisations, particularly those associated with reflexivisation, to be stated easily and transparently.

Tree adjoining grammars (TAGs) (Joshi, Levy and Takahashi, 1975, Vijay-Shanker and Joshi, 1988) possess trees as basic grammar structures, and grammar rules are used to alter the structure of these trees. The relationship between TUG and TAG is very superficial as will be illustrated after a short description of the framework. A TAG contains *initial* trees and *auxiliary* trees. Initial trees are defined as n-ary trees possessing only terminal symbols as leaves. The leaves of an auxiliary tree are all terminal symbols except for a single nonterminal, the *foot*, which is of the same category as the *root* of the tree. These two types of trees comprise the class of elementary trees. There is a tree adjoining operation which is used to form *derived* trees. Application of this rule results in the insertion of auxiliary trees into the middle of initial trees or other derived trees, subject to specific restrictions. TAGs are fundamentally different from TUGs since the adjoining operation alters the structure of the tree instead of merely further instantiating it. Adjoining involves the insertion of trees at internal nodes while the TUG operation can be viewed as the overlaying of trees to form larger structures. The TAG framework has fully specified trees that are modified by other fully specified trees in order to obtain more complex fully specified trees. In TUG, partially specified trees are combined (not modified) in order to obtain a more fully specified complex tree. Feature structure based TAGs (FTAGs) (Vijay-Shanker and Joshi, 1988) are more closely related to TUG than traditional TAGs. The adjoining operation of FTAG amounts to combining a description of the auxiliary tree with that of the tree into which it is adjoined. In this way, a more complete description of the final tree is gradually constructed. However, in FTAG tree descriptions the internal tree structure is not fixed. The descriptions are organised so that additional trees may be adjoined at specific locations. After all the required adjoining operations have been performed, these gaps in the tree structure are closed via unification. In TUG tree descriptions (FA specifications) the internal tree structure is fixed; the fringe nodes of the FA specification are the only ones for which tree structure information may not be specified (as designated by the *hanging edges* described earlier).

The most closely related grammar formalism to TUG is HPSG as described in (Pollard and Sag, 1987). The phrasal signs of HPSG are almost notational variants of the FA specifications of TUG; phrasal signs were not present in the early forms of HPSG (Pollard, 1985) from which UCG and TUG evolved. Aside from the slightly different appearance of these different structures, FA specifications are slightly more restrictive in that a node may only have two descendents instead of the unlimited number allowed in HPSG. TUG also differs from HPSG in that it requires only one (instead of two) grammar rules. This is a consequence of TUG having essentially phrasal-signs as lexical entries. In this way, a lexical entry can directly access information other than that associated with its sister signs in a derivation tree (or phrasal sign). This allows interesting proposals for the treatment of reflexives in controlled complements and unbounded dependency constructions which are discussed in detail in (Popowich, 1988).

SUMMARY

In TUG, the phonological, syntactic, semantic and antecedent information describing linguistic expressions is contained in signs which are organised into FA structures. These FA structures are binary trees which encode the functor-argument dependencies between the signs corresponding to components of a complex expression. Partial specifications of FA structures are associated with individual lexical entries and these FA specifications are combined by a single grammar rule. Dependencies between information associated with different linguistic constituents that are traditionally captured by grammar rules are captured explicitly in the TUG lexical entries. TUG can in some sense be viewed as a 'lexicalised' UCG, where 'lexicalised' is used in the sense discussed in (Schabes, Abeille and Joshi, 1988).

However, the FA structures described by a TUG analysis of a sentence are difficult to obtain as derivation trees in UCG. As discussed earlier, the UCG grammar rules require the semantic attributes of the root-sign and functor-sign of any subtree to be the same. Additional grammar rules would be needed by UCG to allow the different relationships between semantic information

and to allow the three different relations between the R-antecedent information of a root-sign and functor-sign. The R-antecedent information of a functor-sign can either be the same as that of the root-sign (non-predicative functors), or it can consist of the semantic index of its argument in addition to the R-antecedent information of the root-sign (predicative functors), or it can contain only the semantic index of its argument (generalised predicative functors).

The R-antecedent information contained in FA specifications is treated on a level equal to the other forms of information; there is no need to invoke special mechanisms for passing this information. Its distribution is governed by the predication command and generalised predicative constraints. The reflexive attribute of the sign contains information that *might* be needed by a reflexive pronoun. So if a sign for a reflexive pronoun appears in an FA specification, the possible antecedents for the reflexive are easily accessible. During tree unification, if the sign associated with a reflexive pronoun contains no variables of the appropriate sort in its reflexive store, then the use of the pronoun is ungrammatical and tree unification fails. Since an FA specification is associated with each potential antecedent of a reflexive pronoun, failure of anaphora resolution can constrain possible analyses; if there is no possible antecedent for a reflexive, there will not be an FA specification.

REFERENCES

Bach, Emmon, and Barbara Partee. (1980). Anaphora and Semantic Structure. In C. Masek, P. Hendrick and M. Miller (Eds.), *Papers from the Parasession on Language and Behavior at the 17th Regional Meeting of the Chicago Linguistics Society*. Chicago, IL.

Bouma, Gosse. (1988). Modifiers and Specifiers in Categorial Unification Grammar. *Linguistics*, *26*(1), 21-46.

Bouma, Gosse, Ester Koenig, and Hans Uszkoreit. (1988). A Flexible Graph-Unification Formalism and its Application to Natural Language Processing. In *IBM Journal of Research and Development*. Special Issue on Computational Linguistics.

Chierchia, Gennaro. (1988). Aspects of a Categorial Theory of Binding. In R. Oehrle, E. Bach, and D. Wheeler (Eds.), *Categorial Grammars and Natural Language Structures*. D. Reidel, Dordrecht, Holland.

Cooper, Robin. (1983). *Quantification and Syntactic Theory*. D. Reidel, Dordrecht, Holland.

Dowty, David, Robert Wall, and Stanley Peters. (1981). *Introduction to Montague Semantics*. D. Reidel, Dordrecht, Holland.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. (1985). *Generalized Phrase Structure Grammar*. Basil Blackwell, London.

Hellan, Lars. (1988). *Anaphora in Norwegian and the Theory of Grammar*. Foris Publications, Dordrecht, Holland.

Jackendoff, Ray. (1977). *X-bar Syntax: A Study of Phrase Structure*. MIT Press, Cambridge, MA.

Johnson, Mark. (1987). *Attribute-Value Logic and the Theory of Grammar*. Doctoral dissertation, Department of Linguistics, Stanford University, CA.

Johnson, Mark, and Ewan Klein. (1986). Discourse, Anaphora and Parsing. In: *11th International Conference on Computational Linguistics*. Bonn University, West Germany.

Joshi, Aravind, Leon Levy, and M. Takahashi. (1975). Tree Adjunct Grammars. *J. Comput. Syst. Sci.*, Vol. *10*(1).

Kamp, Hans. (1981). A Theory of Truth and Semantic Representation. In J. Groenendijk, T. Janssen, and M. Stokhof (Eds.), *Formal Methods in the Study of Language*. Mathematical Centre Tracts, Amsterdam.

Kaplan, Ron, and Joan Bresnan. (1982). Lexical-Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Kaplan, Ron, John Maxwell, and Annie Zaenen. (January 1987). Functional Uncertainty. In: *The CSLI Monthly*, Centre for the Study of Language and Information, Stanford University, CA.

Kasper, Robert, and William Rounds. (1986). A Logical Semantics for Feature Structures. In: *24th meeting Assoc. Comput. Ling.* Columbia University, New York, N.Y.

Keenan, Edward. (1974). The Functional Principle: Generalizing the Notion of 'Subject of'. In M. La Galy, R. Fox, and A. Bruck (Eds.), *Papers from the 10th Regional Meeting of the Chicago Linguistics Society*. Chicago, IL.

Pollard, Carl. (1985). Lectures on HPSG. Unpublished lecture notes, CSLI, Stanford University, CA.

Pollard, Carl, and Ivan Sag. (1983). Reflexives and Reciprocals in English: An Alternative to the Binding Theory. In M. Barlow, D. Flickinger, and M. Westcoat (Eds.), *Proceedings of the 2nd West Coast Conference on Formal Linguistics*. Stanford Linguistics Association, Stanford, CA.

Pollard, Carl, and Ivan Sag. (1987). *Information-Based Syntax and Semantics, Report 1: Fundamentals*. Centre for the Study of Language and Information, Stanford University, CA.

Popowich, Fred. (1988). *Reflexives and Tree Unification Grammar*. Doctoral dissertation, Centre for Cognitive Science, University of Edinburgh, Edinburgh, Scotland.

Schabes, Yves, Anne Abeille, and Aravind Joshi. (1988). Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. In: *12th International Conference on Computational Linguistics*. Budapest, Hungary.

Shieber, Stuart, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and M. Tyson. (1983). The Formalism and Implementation of PATR-II. In B. Grosz and M. Stickel (Eds.), *Research on Interactive Acquisition and Use of Knowledge*. SRI International, Menlo Park, CA.

Uszkoreit, Hans. (1986). Categorial Unification Grammars. In: *11th International Conference on Computational Linguistics*. Bonn University, West Germany.

van Benthem, Johan. (1987). Categorial Equations. In E. Klein and J. van Benthem (Eds.), *Categories, Polymorphism and Unification*. Centre for Cognitive Science, University of Edinburgh, and Institute for Language, Logic and Information, University of Amsterdam.

Vijay-Shanker, K., and Aravind Joshi. (1988). Feature Structures Based Tree Adjoining Grammars. In: *12th International Conference on Computational Linguistics*. Budapest, Hungary.

Zeevat, Henk, Ewan Klein, and Jo Calder. (1987). An Introduction to Unification Categorial Grammar. In N. Haddock, E. Klein, and G. Morrill (Eds.), *Edinburgh Working Papers in Cognitive Science, Vol.1: Categorial Grammar, Unification Grammar, and Parsing*. Centre for Cognitive Science, Univ. of Edinburgh, Scotland.