A PROLOG IMPLEMENTATION OF LEXICAL FUNCTIONAL GRAMMAR
AS A BASE FOR A NATURAL LANGUAGE PROCESSING SYSTEM

Werner Frey and Uwe Reyle
Department of Linguistics
University of Stuttgart
W-Germany

0. ABSTRACT
The aim of this paper is to present parts of our system [2], which is to construct a database out of a narrative natural language text. We think the parts are of interest in their own. The paper consists of three sections:

(I) We give a detailed description of the PROLOG - implementation of the parser which is based on the theory of lexical functional grammar (LFG). The parser covers the fragment described in [1,4]. I.e., it is able to analyse constructions involving functional control and long distance dependencies. We will to show that PROLOG provides an efficient tool for LFG-implementation: a phrase structure rule annotated with functional schemata like S NP VP --> NP VP will be interpreted as, first, identifying the special grammatical relation of subject position of any sentence analysed by this clause to be the NP appearing in it, and second, as identifying all grammatical relations of the sentence with those of the VP. This universal interpretation of the LFG-metavariables \( \uparrow \) and \( \downarrow \) corresponds to the universal quantification of variables appearing in PROLOG-clauses. The procedural semantics of PROLOG is such that the instantiation of the variables in a clause is inherited from the instantiation given by its subgoals, if they succeed. Thus there is no need for a separate component which solves the set of equations obtained by applying the LFG algorithm.

(II) For the semantic representation of texts we use the Discourse Representation Theory developed by Hans Kamp. At present the implementation includes the fragment described in [4]. In addition it analyses different types of negation and contains equi- and raising-verbs. We postulate some requirements a semantic representation has to fulfill in order to be able to analyse whole texts. We show how Kamp's theory meets these requirements by analyzing sample discourses involving anaphoric NPs.

(III) Finally we sketch how the parser formalism can be augmented to yield as output discourse representation structures. To do this we introduce the notion of 'logical head' in addition to the LFG notion of 'grammatical head'. The reason is the well-known fact that the logical structure of a sentence is induced by the determiners and not by the verb which on the other hand determines the thematic structure of the sentence. However the verb is able to restrict quantifier scope ambiguities or to induce a preference ordering on the set of possible quantifier scope relations. Therefore there must be an interaction between the grammatical head and the logical head of a phrase.

1. A PROLOG IMPLEMENTATION OF LFG
A main topic in AI research is the interaction between different components of a system. But insights in this field are primarily reached by experience in constructing a complex system. Right from the beginning, however, one should choose formalisms which are suitable for a simple and transparent transportation of information. We think LFG meets this requirement. The formalism exhibiting the analysis of a sentence can be expanded in a simple way to contain entries which are used during the parse of a whole text, for example discourse features like topic or domain dependent knowledge coming from a database associated with the lexicon. Since LFG is a kind of unification grammar it allows for constructing patterns which enable the following sentences to refine or to change the content of these discourse features. Knowledge gathered by a preceding sentence can be used to lead the search in the lexicon by demanding that certain feature values match.

In short we hope that the nearly uniform status of the different description tools allows simple procedures for the expansion and manipulation by other components of the system. But this was a look ahead. Let us now come to the less ambitious task of implementing the grammar of [1,4].

Lexical functional grammar (LFG) is a theory that extends phrase structure grammars without using transformations. It emphasizes the role of the grammatical functions and of the lexicon. Another powerful formalism for describing natural languages follows from a method of expressing grammars in logic called definite clause grammars (DCG). A DCG constitutes a PROLOG programme.

We want to show first, how LFG can be translated into DCG and second, that PROLOG provides an efficient tool for LFG-implementation in that it allows for the construction of functional structures directly during the parsing process. I.e. it is not necessary to have separate components which first derive a set of functional equations from the parse tree and then generate an f-structure by solving these equations. Let us look at an example to see how the LFG machinery works. We take as the sample sentence 'a woman expects an american to win'. The parsing of the sentence proceeds along the following lines. The phrase structure rules in (1) generate the phrase structure tree in (2) (without considering the schemata written beneath the rule elements).

(1) S --> NP VP
    NP --> (PSBJ)=\( \uparrow \) \( \downarrow \) (TENSE)
    VP --> V NP NP PP VP'
    VP' --> (to) VP
    NP --> DET N
    \( \uparrow \) \( \downarrow \)

(2) a\( \uparrow \) woman\( \downarrow \) expects\( \uparrow \) an\( \downarrow \) american\( \uparrow \) to\( \downarrow \) win

Then the c-structure will be annotated with the functional schemata associated with the rules. The schemata found in the lexical entries are attached to the leave nodes of the tree. This is shown in (3).

This work is supported by DFG RO 245/13
The tree will be indexed. The indices instantiate the up-arrows. An up-arrow refers to the node dominating the feature symbols. The crucial elements of LFG (in contrast to grammatical schemata) can specify no more than two iterated feature applications.

The equation makes it also possible that a lexical item, here the verb, can induce a functional control relationship between different f-structures of the sentence. An important constraint for all references to functions and functional features is the principle of functional locality: designators in lexical and grammatical schemes can specify no more than two iterated function applications.

Why is especially PROLOG useful for doing this? In the annotated c-structure of the LFG theory the content of the functional equations is only "known" by the node the equation is annotated to and by the immediately dominating node. The memory is so to speak locally restricted. Thus during the parse all those bits of information have to be protocolled for some other nodes. This is done by means of the equations. In a PROLOG programme the nodes turn into predicates with arguments. The arguments could be the same for different subgoals within a clause. Therefore the memory is "horizontally" not restricted at all. Furthermore by sharing of variables the predicates which are goals can give information to their subgoals. In short, once a phrase structure grammar has been translated into a PROLOG programme every node is potentially able to grasp information from any other node.

The parser we get by embedding the restricted LFG formalism into the highly flexible DEF grammar formalism respects the constraints of Lexical functional grammar. Another important fact is that LFG calls the PROLOG programmer in an exact manner what information the parser needs at which node and just because this information is purely locally represented in the LFG formalism it leads to the possibility of translating LFG into a PROLOG programme in a canonical way. We have said that in solving the equations DEF sticks together informations coming from different nodes to build up the final output. To mirror this the following PROLOG feature is of greatest importance. For the construction of the wanted output the parser sets the desired output during the parsing process structures can be built up piecewise, leaving unspecified parts as variables. The construction of the output need not be strictly parallel to the application of the corresponding rules. Variables play the role of placeholders for structures which are found possibly later in the parsing process. A closer look at the verb entries as formulated by LFG reveals that the role of the function names appearing there is to function as placeholders too.

To summarize: By embedding the restricted LFG formalism into the highly flexible definite clause grammar formalism we make life easier. Nonetheless the parser get respects the constraints which are formulated by the LFG theory.

Let us now consider some of the details. The rules under (1)
We use the content of the function assigning equations to build the output of the subprocedure corresponding to the head. The head category of a phrase is characterized by the assignment of the major category, called its head, with the property that the functional control is done by the indicated f-structures. For each category there is a fixed set of features, the head category is able to impose restrictions on a fixed subset of that feature set. This subset is placed on a prominent position. The corresponding feature values percolating up towards the head category will end up in the same position demanding that their values agree. This is done by the feature variables. The uniqueness condition is trivially fulfilled since the passing around of parts of the f-structure is done by variables, and PROLOG instantiates a variable with at most one value.

(7) V (*XOMP (SUBJ *outpobj) #featobj)) [functional control] (SUBJ (*outpobj) (SG 3)) \xrightarrow{\text{check list}} (SUBJ (*outpobj) #featobj) (XOMP (outpcomp)) TEN

The checking of the completeness and coherence condition is done by the Verb procedure. (7) shows the PROLOG assertion corresponding to the lexical entry for "expects." In every assertion for verbs there is a list containing the grammatical functions subcategorized by the verb. This is the second argument in (7), called 'check list'. This list is passed around during the parse. This is done by the list underlined with waves in (6). Every subcategorized function appearing in the sentence must be able to shorten the list. This guarantees coherence. In the end the list must have diminished to NIL. This guarantees completeness.

As can be seen in (7) a by-product of this passing around the check list is to bring the values of the grammatical functions subcategorized by the verb down to the verb's predicate argument structure.

To handle functional control the verb entry contains an argument to encode the controller. This is the first argument in (7). The procedure which delivers XOMP (here the VP procedure) receives this variable (the underlined variable *cont in (6)) since verbs can induce functional control only upon the open grammatical function XOMP. For tough-movement constructions the s-prime procedure receives the controller variable too. But inside this clause the controller must be put onto the long distance controller list, since XOMP is not an open grammatical function. This leads us to the long distance dependencies.

(8) The girl wondered whose playmate's nurse the baby saw.
(9) S' --> NP VP

(10) NP --> DET NP NP

In English questions and relatives an element at the front of the clause is understood as filling a particular grammatical role within the clause, determined by the position of a c-structure gap. Consider sentence (8). This kind of dependency is called constituent control, because in contrast to functional control the constituent structure configurations are the primary conditioning factors and not lexical items. Bresnan/Kaplan introduce a new formal mechanism for representing long-distance dependencies. To handle the embedded question sentence they use the rule in (9). The double arrow downwards represents the controller of the constituent control relationship. To this arrow corresponds another double arrow which points upwards and represents the controller. This one is attached for example to the empty string NP --> . But as the arrow indexed with [+wh] shows the controller may also be a designated set of lexical items which include interrogative pronouns, determiners and adverbs. "whose" for example has the lexical entry: whose N, (FRED) = who", OSE = GEN, F = . (This kind of control relationship is needed to analyze the complex NP "whose playmate's nurse the baby saw." in sentence (8)). The control relationships are illustrated in (10). Corresponding controllers and controllers must have compatible subscripts. The subscripts indicate the category of the controller. The superscript S of the one controller indicates that the corresponding controller has to be found in a S-rooted control domain whereas the [+wh] controller for the other controller has to be found beneath a NP node. Finally the box around the S-node needs to be explained. It indicates the fact that the node is a bounding node. Kaplan/Bresnan state the following convention: A node M belongs to a control domain with root node R if and only if R dominates M and there are no bounding nodes on the path from M up to but not including R. This convention prevents constructions like the one in (11).

(11) The girl wondered what the nurse asked who saw the baby.

This convention prevents constructions like the one in (11).
controller variable of the $S'$ procedure in (12). \( \gamma \) is the output variable. \((\gammaLd)\) is expanded by the \((\gammaLd)\) controller within the NP subgoal. This controller must find its controllee during the execution of the NP goal. Note that the output variable of the NP subgoal is identical with the output variable of the main goal and that the subgoal \( S' \) does have different controller lists. This reflects the effect of the box around the \( S \)-node, i.e. no controller coming downwards can find its controllee within the \( S \)-procedure. The only controller going into the \( S \) goal is the one introduced below the NP node with domain root \( S \). Clearly the output variable of \( S \) has to be \( \gamma \).

There are rules which allow for certain controllers to pass a boxed node Bransen/Kaplan state for example the rule in (13).

\[
\begin{align*}
S' & \rightarrow (\text{that}) \ S \\
\end{align*}
\]

This rule has the effect that \( S \)-rooted controllers are allowed to pass the box. Here we use a test procedure which puts only the controllers indexed by \( S \) onto the controller list going to the \( S \) goal. Thereby we obtain the right treatment of sentence (14). (14) The girl wondered who John believed that Mary claimed that the baby saw. In a corresponding manner the complex NP 'whose playmate's nurse' of sentence (8) is analysed.

II. SEMANTIC REPRESENTATION

As semantic representation we use the IXRS (representation) T(heory) developed by Hans Kamp [4], i.e. we do not adopt the semantic theory for LF (lexical Functional) G(rammar) proposed by Per-Kristian Halverson [2]. Halverson translates the functional structures of LFG into so-called semantic structures being of the same structural nature, namely acyclic graphs. The semantic structures are the result of a translation procedure which is based on the association of formulas of intensional logic to the semantic forms appearing in the functional structure. The reason not to take this approach will be explained by postulating some requirements a semantic representation has to fulfill in order to account for a processing of texts. Then we will show that these requirements are really necessary by analysing some sample sentences and discourses. It will turn out that IXRS accounts for them in an intuitively fully satisfactory way.

Because we cannot review IXRS in detail here the reader should consult one of the papers explaining the fundamental of its theory (e.g. [1]), or he should first look at the last paragraph in which an outline is given of how our parser is to be extended in order to yield an IXRS-typed output - instead of the 'traditional' (semantic) functional structures.

The basic building principle of a semantic representation is to associate with every significant lexical entry (i.e., every entry which does contribute to the truthconditional aspect of the meaning of a sentence) a semantic structure. Compositional principles, then, will construct the semantic representation of a sentence by combining these semantic structures according to their syntactic relations. The desired underlying principle is that the semantic structures associated with the semantic forms should not be changed during the composition process. To put it differently: one wants the association of the semantic structures to be independent of the syntactic context in which the semantic forms appear. This requirement leads to difficulties in the tradition of translating sentences into formulas of e.g. predicate or intensional logic.

Consider sentences

(1) If John admires a woman then he kisses her and
(2) Every man who admires a woman kisses her
(3) \( \forall x (\text{woman}(x) \land \text{admirer}(John,x) \rightarrow \text{kiss}(John,x)) \)

and

(4) \( \forall x \forall y (\text{man}(x) \land \text{woman}(y) \land \text{admire}(x,y) \rightarrow \text{kiss}(x,y)) \)

respectively. The problem is that the definite description "a woman" reemerges as universally quantified in the logical representation - and there is no way out, because the pronoun "she" has to be bound to the woman in question. IXRS provides a general account of the meaning of indefinite descriptions, complex ones, universally quantified noun phrases and anaphoric pronouns, s.t. our first requirement is satisfied. The semantic representations (called IXRS's) which are assigned to sentences in which such constructions jointly appear have the truth conditions which our intuitions attribute to them.

The second reason why we decided to use IXRS as semantic formalism for LFG is that the construction principles for a sentence \( S(1), \ldots, S(n) \) are formulated with respect to the semantic representation of the preceding text \( S(1), \ldots, S(i-1) \). Therefore the theory can account for intersentential semantic relationships in the same way as for intrasentential ones. This is the second requirement: a semantic representation has to represent the discourse as a whole and not as the mere union of the semantic representations of its isolated sentences.

A third requirement a semantic representation has to fulfill is the reflection of configurational restrictions on anaphoric links: If one embeds sentence (2) into a conditional

(6) "If every man who admires a woman kisses her then she is happy."

one has to take into account some knowledge of the world, namely the fact that every car has exactly one engine.

Because we cannot review IXRS in detail here the reader should consult one of the papers explaining the fundamental of its theory (e.g. [1]), or he should first look at the last paragraph in which an outline is given of how our parser is to be extended in order to yield an IXRS-typed output - instead of the 'traditional' (semantic) functional structures.

A last requirement we will stipulate here is that the semantic representation has to represent the discourse as a whole and not as the mere union of the semantic representations of its isolated sentences.

Consider

(8) Pedro is a farmer. If a woman loves him then he is happy.

Mary loves Pedro. The happy farmer marries her and

(9) John bought a car. The engine has 160 horse powers

Pedro is a farmer. If a woman loves him then he is happy.

Mary loves Pedro. The happy farmer marries her

Thus an IXRS consists of

(1) a set of discourse referents: discourse individuals, discourse events, discourse propositions, etc.
(2) a set of conditions of the following types

- atomic conditions, i.e. \( \land, \lor \) over discourse referents
- complex conditions, i.e. \( \land, \lor \) over sub-IXRS's and discourse referents (e.g. \( \forall (x) \rightarrow (y) \) or
must be such that it allows for a proper embedding of K in it.

The truth condition states that a IRS K is true in a model M if there is a proper embedding from K into M. Proper embedding is defined as a function f from the set of discourse referents of K in to M s.t. (1) it is a homomorphism for the atomic conditions of the IRS and (ii) - for the case of a complex condition K(1) → K(2) every proper embedding of K(1) that extends f is extendable to a proper embedding of K(2).

For the case of a complex condition pK the modal-theoretic object correlated with p (i.e. a proposition if p is a discourse proposition, an event if p is a discourse event, etc.) must be such that it allows for a proper embedding of K in it. Note that the definition of proper embedding has to be made more precise in order to adapt it to the special semantics one uses for propositional attitudes. We cannot go into details here. Nonetheless the truth condition as it stands should make clear the following: whether a discourse referent introduced implies existence or not depends on its position in the hierarchy of the IRS’s. Given a IRS which is true in M then exactly those referents introduced in the very top-level IRS imply existence; all others are to be interpreted as universally quantified, if they occur in an antecedent IRS, or as existentially quantified if they occur in a consequent IRS, or as having opaque status if they occur in a IRS specified by e.g. a discourse proposition. Thus the role of the hierarchical order of the IRS’s is to build a base for the definition of truth conditions. But furthermore the hierarchy defines an accessibility relation, which restricts the set of possible antecedents of anaphoric NP’s. This accessibility relation is (for the fragment in (4)) defined as follows:

For a given sub-IRS TD all referents occurring in TD or in any of the IRS’s in which f is embedded are accessible. Furthermore if TD is a consequent-IRS then the referents occurring in its corresponding antecedent IRS on the left are accessible too.

This gives us a correct treatment for (6) and (7).

For the time being - we have no algorithm which restricts and orders the set of possible anaphoric antecedents according to contextual conditions as given by e.g. (5).

(5) John is reading a book on syntax and Bill is reading a book on semantics.

The former is enjoying himself

{is a paperback}

Therefore our selection set is restricted only by the accessibility relation and the descriptive content of the anaphoric NP’s. Of course for anaphoric pronouns this content is reduced to a minimum, namely the grammatical features associated to them by the lexical entries. This accounts e.g. for the difference in acceptability of (10) and (11).

(10) Mary persuaded every man to shave himself

(11) Mary persuaded every man to shave himself

The IRS’s for (10) and (11) show that both discourse referents, the one for "Mary" and the one for a "man", are accessible from the position at which the reflexive pronoun has to be resolved. But if the "himsell" of (11) is replaced by x it cannot be identified with y having the (not explicitly shown) feature female.

\[
\begin{align*}
&y \\
&\text{may} \\
&x \\
&\text{promise(y,x,p)} \\
&\text{promise(y,p,x)} \\
&p: \\
&\text{man(x)} \\
&\text{shave(x,x)} \\
&\text{shave(y,\text{himself})}
\end{align*}
\]

Definite descriptions bear more information by virtue of the semantic content of their common-noun-phrases and the existence and uniqueness conditions presupposed by them. Therefore in order to analyse definite descriptions we look for a discourse referent introduced in the preceding IRS for which the description holds and we have to check whether this description holds for one referent only. Our algorithm proceeds as follows:

First we build up a small IRS KD encoding the descriptive content of the common-noun-phrase of the definite description together with its uniqueness and existency condition:

\[
\begin{align*}
&\text{KD:} \\
&\text{x} \\
&\text{farmer(x)} \\
&\text{happy(x)} \\
&\text{farmer(y)} \\
&\text{happy(y)} \\
&\rightarrow y = x
\end{align*}
\]

Second we have to show that we can prove KD out of the text-IRS of the preceding discourse, with the restriction that only accessible referents are taken into account. The instantiation of *x by this proof gives us the correct antecedent the definite description refers to. Now we forget about KD and replace the antecedent discourse referent for the definite noun phrase to get the whole text-IRS (8').

Of course it is possible that the presuppositions are not mentioned explicitly in the discourse but follow implicitly from the text alone or from the text together with the knowledge of the domain it talks about. So in cases like (9) John bought a car. The engine has 260 horse powers

Here the identified referent is functionally related to referents that are more directly accessible, namely to John’s car. Furthermore such a functional dependency confers to a definite description the power of introducing a new discourse referent, namely the engine which is functionally determined by the car of which it is part. This shifts the task from the search for the direct antecedent for "the engine" to the search for the referent it is functionally related to. But the basic mechanism for finding this referent is the same deductive mechanism just outlined for the "happy farmer" example.

III. TOWARDS AN INTERACTION BETWEEN "GRAMMATICAL PARSING" AND "LOGICAL PARSING"

In this section we will outline the principles underlying the extension of our parser to produce IRS’s as output. Because none of the fragments of IR contain raising- and equal-verbs taking infinitival or that-complements we are confronted with the task of writing construction rules for such verbs. It will turn out, however, that it is not difficult to see how to extend IR to comprise such constructions. This is due to the fact that using LPC as syntactic base for IR - and not the categorial syntax of Kemp - the unraveling of the thematic relations in a sentence is already accomplished in f-structure. Therefore it is straightforward to formulate construction rules which give the correct readings for (10) and (11) of the previous section, establish the propositional equivalence of pairs with or without raising, equal (see (1), (2)), etc.

(1) John persuaded Mary to come

(2) John persuaded Mary that she should come

Let us first describe the IRS construction rules by the familiar example

(3) Every man loves a woman

Using Kemp’s categorial syntax, the construction rules operate top down the tree. The specification of the order in which the parts of the tree are to be treated is assumed to be given by the syntactic rules. I.e. the specification of scope order is directly determined by the syntactic construction of the sentence. We will deal with the point of scope ambiguities after having described the way a IRS is constructed. Our description - operating bottom up instead top down - is different from the one given in [4] in order to come closer to the point we want to make. But note that this difference is not a genuine one. Thus according to the first requirement of the previous section we assume that to each IRS an f-structure is associated. For the lexical entries of (3) we have
Of course, in order to make the definition work it has to be clarified that the logical head of a phrase is uniquely determined too. Consider (7) John expected an American to win

The semantic structures for the common nouns and the verbs are n-place predicates. The structure for "a" is a BRS with discourse individual v introduced and conditions not yet specified. The entry for "every" is a BRS with no discourse individual introduced on the toplevel. It contains however a complex condition K1 \rightarrow K1 s.t. a discourse individual x is introduced in K1 and both K0 and K1 contain any other conditions.

The BRS construction rules specify how these semantic structures are to be generated by propagating them up the tree. The easiest way to illustrate that is to do it by the following picture (for the case of narrow scope reading of "a woman"):

(4) \begin{center}
\begin{tikzpicture}[scale=0.6]
  \node {\texttt{NP:}} at (0,0) [rectangle, draw]
  \node {\texttt{man(x)}} at (0,1.5)
  \node {\texttt{woman(v)}} at (1.5,1.5)
  \node {\texttt{VP:}} at (3,0) [rectangle, draw]
  \node {\texttt{V}} at (3,1.5)
  \node {\texttt{loves(*,v)}} at (4.5,1.5)
\end{tikzpicture}
\end{center}

For the wide scope reading the NP-tree of "a woman" is treated at the very end to give (5)

(5) \begin{center}
\begin{tikzpicture}[scale=0.6]
  \node {\texttt{NP:}} at (0,0) [rectangle, draw]
  \node {\texttt{man(x)}} at (0,1.5)
  \node {\texttt{love(*,v)}} at (1.5,1.5)
  \node {\texttt{VP:}} at (3,0) [rectangle, draw]
  \node {\texttt{V}} at (3,1.5)
  \node {\texttt{woman(v)}} at (4.5,1.5)
\end{tikzpicture}
\end{center}

The picture should make clear the way we want to extend the parsing mechanism described in section 1 in order to produce BRS's as output and no more f-structures: instead of partially instantiated f-structures determined by the lexical entries partially instantiated BRS's are passed around the tree getting accomplished by unification. The control mechanism of LFG will automatically put the discourse referents into the correct argument position of the verb. Thus no additional work has to be done for the grammatical relations of a sentence.

But what about the logical relations?

Recall that each phrase has a unique head and that the functional features of each phrase are identified with those of its head. For (3) the head of S \rightarrow NP VP IS the VP and the head of VP \rightarrow V NP IS the V. Thus the outstanding role of the verb to determine and restrict the grammatical relations of the sentence is captured. (4) , however, shows that the grammatical relations of the sentence are mainly determined by its determiners, which are not heads of the NP-phrases and the NP-phrases themselves are not the heads of the VP- and S-phrase respectively. To account for this dichotomy we will call the syntactically defined notion of head "grammatical head" and we will introduce a further notion of "logical head" of a phrase. Of course, in order to make the definition work it has to be elaborated in a way that guarantees that the logical head of a phrase is uniquely determined too. Consider (5) John persuaded an American to win

For which we propose the following BRS's (6)

(6) \begin{center}
\begin{tikzpicture}[scale=0.6]
  \node {\texttt{NP:}} at (0,0) [rectangle, draw]
  \node {\texttt{man(x)}} at (0,1.5)
  \node {\texttt{woman(v)}} at (1.5,1.5)
  \node {\texttt{VP:}} at (3,0) [rectangle, draw]
  \node {\texttt{V}} at (3,1.5)
  \node {\texttt{loves(*,v)}} at (4.5,1.5)
\end{tikzpicture}
\end{center}

The fact that (7) does not necessarily imply existence of an American whereas (6) does is triggered by the difference between Equi- and Raising-verbs.

Suppose we define the NP to be the logical head of the phrase VP \rightarrow V NP VP. Then the logical relations of the VP would be those of the NP. This amounts to incorporating the logical structures of the V and the VP into the logical structure of the NP, which is for both (6) and (7)

\begin{center}
\begin{tikzpicture}[scale=0.6]
  \node {\texttt{John = j}} at (0,0)
  \node {\texttt{america(y)}} at (1,0)
  \node {\texttt{persuade(j,y,p)}} at (2,0)
  \node {\texttt{win(y)}} at (3,0)
\end{tikzpicture}
\end{center}

and thus would lead to the readings represented in (6') and (7'). Consequently (7') would not be produced.

Defining the logical head to be the VP would exclude the readings (6') and (7').

Evidently the last possibility of defining the logical head to be identical to the grammatical head, namely the V itself, seems to be the only solution. But this would block the construction already at the stage of unifying the NP- and VP-structures with persuade(*,*,*) or expect(*,*,*). At first thought one easy way out of this dilemma is to associate with the lexical entry of the verb not the empty n-place predicate but a IRS containing this predicate as atomic condition. This makes the unification possible but gives us the following result:

(7') \begin{center}
\begin{tikzpicture}[scale=0.6]
  \node {\texttt{John = j}} at (0,0)
  \node {\texttt{america(y)}} at (1,0)
  \node {\texttt{persuade(j,y,p)}} at (2,0)
  \node {\texttt{win(y)}} at (3,0)
\end{tikzpicture}
\end{center}

Of course one can say that (8) is open to produce the set of BRS's representing (6) and (7). But this means that one has to work on (8) after having reached the top of the tree - a consequence that seems undesirable to us.

Thus the only way out is to consider the logical head as not being uniquely identified by the mere phrase structure configurations. The above example for the phrase VP \rightarrow V NP VP shows its head depends on the verb class too. But we will still go further.

We claim that it is possible to make the logical head to additionally depend on the order of the surface string, on the use of active and passive voice and probably others. This will give us a preference ordering of the scope ambiguities of sentences as the following:

- Every man loves a woman
- A woman is loved by every man
- A ticket is bought by every man
- Every man bought a ticket

The properties of unification grammars listed above show that the theoretical framework does not impose any restrictions on that plan.

REFERENCES

[1] Bresnan, J. (ed.), "The Mental Representation of Grammatical Relations". MIT Press, Cambridge, Mass., 1982
[2] Frey, Werner/ Reyde, Uwe/ Rohrer, Christian, "Automatic Construction of a Knowledge Base by Analysing Texts in Natural Language", in: Proceedings of the Eight Intern. Joint Conference on Artificial Intelligence II, 1983
[3] Halverson, P.-K., "Semantics for Lexical Functional Grammar", In: Linguistic Inquiry 14, 1982
[4] Kemp, Hans, "A Theory of Truth and Semantic Representation", In: J.A. Groenendijk, T.U.V. (ed.), Formal Semantics in the Study of Natural Language 1, 1981