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

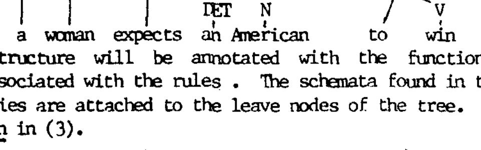
0. ABSTRACT

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 $S \rightarrow \underset{(\uparrow SUBJ)=\downarrow}{NP}\ \underset{\uparrow=\downarrow}{VP}$ is to be interpreted as, first, identifying the special grammatical relation of subject position of any sentence analyzed 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 ↑ and ↓ 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.
-there is a canonical way of translating LFG into a PROLOG programme.
(II) For the semantic representation of texts we use the Discourse Representation Theory developped by Hans Kamp. At present the implementation includes the fragment described in [4]. In addition it analyses different types of negation and certain 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 NP's.
(III) Finally we sketch how the parser formalism can be augmented to yield as output discourse representation structures. To do this we introduce the new notion of 'logical head' in addition to the LFG notion of 'grammatical head'. The reason is the wellknown 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.

I. A PROLOG IMPLEMENTATION OF LFG

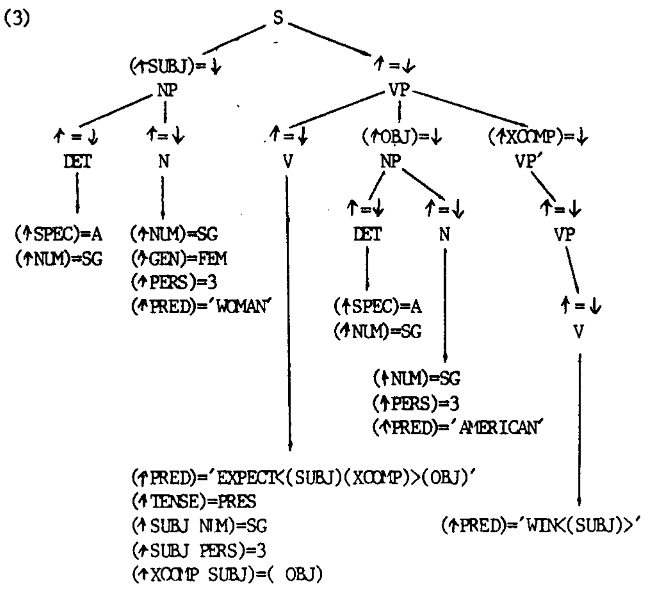
A main topic in AI research is the interaction between different components of a system. But insights in this field are primarily reached by experience in constructing a complex system. Right from the beginning, however, one should choose formalisms which are suitable for a simple and transparent transportion 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 comming 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 seperate components which first derive a set of functional equations from the parse tree and secondly 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)
$$S \longrightarrow \underset{(\uparrow SUBJ)=\downarrow}{NP} \quad \underset{\uparrow=\downarrow}{VP} \quad (\uparrow TENSE)$$
$$VP \longrightarrow \underset{\uparrow=\downarrow}{V} \quad \underset{(\uparrow OBJ)=\downarrow}{NP} \quad \underset{(\uparrow OBJ2)=\downarrow}{NP} \quad \underset{(\uparrow(\downarrow PCASE)=\downarrow}{PP} \quad \underset{(\uparrow XCOMP)=\downarrow}{VP'}$$
$$VP' \longrightarrow (to) \quad \underset{\uparrow=\downarrow}{VP}$$
$$NP \longrightarrow \underset{\uparrow=\downarrow}{DET} \quad \underset{\uparrow=\downarrow}{N}$$

(2)


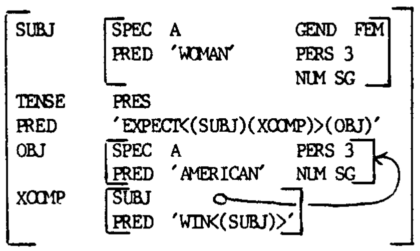
Then the c-structure will be annotated with the functional schemata associated with the rules . The schemata found in the lexical entries are attached to the leave nodes of the tree.
This is shown in (3).

This work is supported by DFG RO 245/13

(3)


(4) (f1 SUBJ) = f2   f3 = f6
f1 = f3   (f6 PRED) = 'EXPECT<(SUBJ)(XCOMP)>(OBJ)'
f2 = f4   (f6 TENSE)= PRES
f2 = f5   (f6 XCOMP SUBJ) = (f6 OBJ)
(f5 NUM) = SG   (f5 PRED) = 'WOMAN'
⋮   ⋮

Then the tree will be indexed. The indices instantiate the up- and down-arrows. An up-arrow refers to the node dominating the node the schema is attached to. A down-arrow refers to the node which carries the functional schema.
The result of the instantiation process is a set of functional equations. We have listed part of them in (4). The solving of these equations yields the functional structure in (5).

(5)


It is composed of grammatical function names, semantic forms and feature symbols. The crucial elements of LFG (in contrast to transformational grammar) are the grammatical functions like SUBJ, OBJ, XCOMP and so on. The functional structure is to be read as containing pointers from the function-names appearing in the semantic forms to the corresponding f-structures.
The grammatical functions assumed by LFG are classified in subcategorizable (or governable) and nonsubcategorizable functions. The subcategorizable ones are those to which lexical items can make reference. The item 'expects' for example subcategorizes three functions, but only the material inside the angled brackets list the predicate's semantic arguments. XCOMP and XADJ are the only open grammatical functions, i.e.,they can denote functionally controlled clauses. In our example this phenomena is lexically induced by the verb 'expects'. This is expressed by its last schema "(↑XCOMP SUBJ)=(↑OBJ)". It has the effect that the OBJ of the sentence will become the SUBJ of the XCOMP, that means in our example it becomes the argument of the predicate 'win'.
Note that the analysis of the sentence 'a woman promises an American to win' would differ in two respects. First the verb 'promises' lists all the three functions subcategorized by it in its semantic argument structure. And second 'promises' differs from 'expects' just in its functional control schema, i.e., here we find the equation "(↑XCOMP SUBJ)=(↑SUBJ)" yielding an arrow from the SUBJ of the XCOMP to the SUBJ of the sentence in the final f-structure.
An f-structure must fulfill the following conditions in order to be a solution
-uniqueness: every f-name which has a value has a unique value
-completeness:the f-structure must contain f-values for all the f-names subcategorized by its predicate
-coherence: all the subcategorizable functions the f-structure contains must be subcategorized by its predicate
The ability of lexical items to determine the features of other items is captured by the trivial equations. They propagate the feature set which is inserted by the lexical item up the tree. For example the features of the verb become features of the VP and the features of the VP become features of S. The uniqueness principle guarantees that any subject that the clause contains will have the features required by the verb. The trivial 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 schemata can specify no more than two iterated function applications.
Our claim is that using DCG as a PROLOG programme the parsing process of a sentence according to the LFG-theory can be done more efficiently by doing all the three steps described above simultaneously.
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 however the nodes turn into predicates with arguments. The arguments could be the same for different predicates 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 pragramme every node is potentially able to grasp information from any other node. Nonetheless the parser we get by embedding the restricted LFG formalism into the highly flexible DCG formalism respects the constraints of Lexical functional grammar.
Another important fact is that LFG tells 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 LFG 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 during the parsing process structures can be built up pieceneal, 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 higly flexible definite clause grammar formalism we make life easier. Nonetheless the parser we get respects the constraints which are formulated by the LFG theory.
Let us now consider some of the details. The rules under (1)

are transformed into the PROLOG programme in (6). (* indicates the variables.)

```
(6) S (*cl0 *cl1 *outps)   <——
                NP (*cl0 *cl2 *featnp *outpnp)
                VP (*cl2 *cl1 (SUBJ (*outpnp *featnp)) TEN *outps)
 VP (*cl0 *cl1 *outpsubj *featv *outps)   <——
                V (*cont (*outpsubj . *10) *featv *outps)
                FACNP (*cl0 *cl2 OBJ *10 *11)
 |functional    FACNP (*cl2 *cl3 OBJ2 *11 *12)
  control|      FACPP (*cl3 *cl4 OBL *12 *13)
                FACVP' (*cl4 *cl1 *cont XCOMP *13 nil)   |checklist|
 FACVP' (*cl0 *cl1 (*gf *cont) *gf ((*gf *outpxcomp) . *10) *10)
                <- VP' (*cl0 *cl1 *cont *outpxcomp)
 NP (*cl0 *cl1 *outpnp)   <——
                DET (*cl0 *cl1 *outpdet)
                N (*outpdet *outpnp)
```

We use the content of the function assigning equations to build up parts of the whole f-structure during the parsing process. Crucial for this is the fact that every phrase has a unique category, called its head, with the property that the functional features of each phrase are identified with those of its head. The head category of a phrase is characterized by the assignment of the trivial functional-equation and by the property of being a major category. The output of each procedure is constructed by the subprocedure corresponding to the head. This means that all information resulting from the other subprocedures is given to that goal. This is done by the 'outp' variables in the programme. Thus the V procedure builds up the f-structure of the sentence. Since VP is the head of the S rule the VP procedure has an argument variable for the SUBJ f-structure. Since V is the head of the VP rule this variable together with the structures coming fom the sister nodes are given to V for the construction of the final output. As a consequence our output does not contain pointers in contrast to Bresnan's output. Rather the argument positions of the predicates are instantiated 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 'feat' 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 ( (VCOMP (SUBJ (*outpobj *featobj)))    |functional control|
      ((SUBJ (*outpsubj (SG 3))) <—— |check list|
                  (OBJ (*outpobj *featobj)) (XCOMP *outpxcomp))
      TEN                                          |output|
      ((TENSE PRES)  (PRED 'EXPECT (*outpsubj *outpxcomp)')) )
```

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 subcategorizable 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.

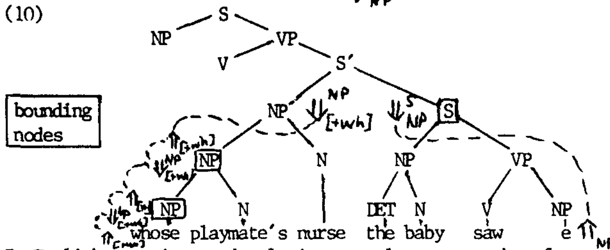
To handle functional control the verb entry contains an argument to encode the controller. This is the first argument in (7). The procedure which delivers XCOMP (here the VP' procedure) receives this variable (the underlined variable *cont in (6)) since verbs can induce functional control only upon the open grammatical function XCOMP. 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 SCOMP is not an open grammatical function.

That leads us to the long distance dependencies

(8) The girl wonders whose playmate's nurse the baby saw ___ .

(9) $S' \longrightarrow \underset{\substack{(\uparrow Q) = \Downarrow^{NP}_{[+wh]} \\ (\uparrow \mathrm{Focus}) = \downarrow \\ \downarrow = \Downarrow^{S}_{NP}}}{NP} \quad \underset{\uparrow = \downarrow}{\boxed{S}}$

(10)



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 controlee. This one is attached for example to the empty string $NP \longrightarrow_{\uparrow} e_{\Uparrow}$. But as the arrow indexed with [+wh] shows the controller may affect also a designated set of lexical items which includes interrogative pronouns , determiners and adverbs. 'whose' for example has the lexical entry: whose N, ($\uparrow$PRED) = 'who', CASE = GEN, $\Uparrow_{[+wh]} = \uparrow$. (This kind of control relationship is needed to analyse the complex NP "whose playmate's nurse" in sentence (8))

The control relationships are illustrated in (10). Corresponding controllers and controlees must have compatible subscripts. The subscripts indicate the category of the controllee. The superscript S of the one controller indicates that the corresponding controlee has to be found in a S-rooted control domain whereas the [+wh] controlee 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 ___

Long distance control is handle by the programme using a long distance controller list, enriched at some special nodes with new controllers, passed down the tree and not allowed to go further at the bounding nodes.

```
(12)    S' (*cl0 *cl1 *outpsc)   <——
  |long              NP (((NP [+wh]) . *cl0) *cl1 *featnp *outpnp)
   distance
   controller        rest (*cl1 *cl0)
   list|             S ((*outpnp *featnp (S NP)) nil *outpsc)
```

Every time a controlee is found its subscript has to match the corresponding entry of the first member of the controller list. If this happens the first element will be deleted from the list. The fact that a controlee can only match the first element reflects the crossed dependency constraint. *cl0 is the input controller variable of the S′ procedure in (12). *cl1 is the output variable. *cl0 is expanded by the [+wh] 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 controlee inside 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 nil. There are rules which allow for certain controllers to pass a boxed node Bresnan/Kaplan state for example the rule in (13).

(13) S′ —> (that) S

$$\begin{array}{c} \uparrow = \downarrow \\ \Uparrow = \Downarrow^{S} \end{array}$$

This rule has the effect that S-rooted contollers are allowed to pass the box. Here we use a test procedure which puts only the contollers 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 D(iscourse) R(epresentation) T(heory) developped by Hans Kamp [4]. I.e. we do not adopt the semantic theory for L(exical) F(unctional) 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 DRT accounts for them in an intuitively fully satisfactory way.

Because we cannot review DRT in detail here the reader should consult one of the papers explaining the fundamentals of the theory (e.g. [4]), 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 DRS-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 truthcondidtional 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 form appears. This requirement leads to difficulties in the tradition of translating sentences into formulas of e.g. predicate or intentional logic.

Consider sentences

(1) If John admires a woman then he kisses her

and

(2) Every man who admires a woman kisses her

the truth conditions of which are determined by the first order formulae

(3) $\forall x$ (woman(x) & admire(John,x) —> kiss(John,x))

and

(4) $\forall x \forall y$ (man(x) & woman(y) & admire(x,y) —> 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. DRT provides a general account of the meaning of indefinite descriptions, conditionals, universally quantified noun phrases and anaphoric pronouns, s.t. our first requirement is satisfied. The semantic representations (called DRS'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 DRR as semantic formalism for LFG is that the construction principles for a sentence S(i) of a text D = S(1),...,S(n) are formulated with respect to the semantic representation of the preceeding text S(1),...,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 stressed

the anaphoric link in (2) is preserved. But (6) does - for configurational reasons - not allow for an anaphoric relation between the "she" and "a woman". The same happens intersententially as shown by

(7) If John admires a woman then he kisses her. *She is enraged.

A last requirement we will stipulate here is the following: It is neccessary to draw inferences already during the construction of the semantic representation of a sentence S(i) of the discourse. The inferences must operate on the semantic representation of the already analyzed discourse S(1),...,S(i-1) as well as on a database containing the knowledge the text talks about. This requirement is of major importance for the analysis of definite descriptions. Consider

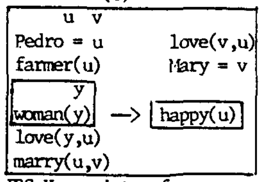
(8) Pedro is a farmer. If a woman loves him then he is happy. Mary loves Pedro. The happy farmer marries her

in which the definite description "the happy farmer" is used to refer to refer to the individual denoted by "Pedro". In order to get this link one has to infer that Pedro is indeed a happy farmer and that he is the only one. If this were not the case the use of the definite description would not be appropriate. Such a deduction mechanism is also needed to analyse sentence

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

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

To illustrate the way the semantic representation has to be interpreted let us have a brief look at the text-DRS for the sample discourse (8)



Thus a DRS K consists of

(i) a set of discourse referents: discourse individuals, discourse events, discourse propositions, etc.

(ii) a set of conditions of the following types

- atomic conditions, i.e. n-ary relations over discourse referents

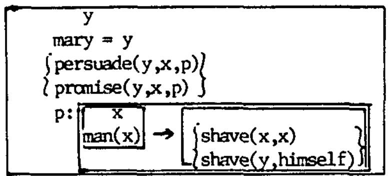
- complex conditions, i.e. n-ary relations (e.g. —> or :) over sub-DRS's and discourse referents (e.g. K(1) —> K(2) or

p:K, where p is a discourse proposition)
A whole IRS can be understood as partial model representing the individuals introduced by the discourse as well as the facts and rules those individuals are subject to.
The truth conditions state that a IRS K is true in a model M if there is a proper imbedding 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. (i) 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 p:K the modeltheoretic 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 eactly those referents introduced in the very toplevel 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 accessibiltity relation is (for the fragment in [4]) defined as follows:
For a given sub-IRS KO all referents occurring in KO or in any of the IRS's in which KO is embedded are accessible. Furthermore if KO 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) John is reading a book on syntax and Bill is reading a book on semantics.

The former $\begin{Bmatrix}\text{is enjoying himself}\\ \text{is a paperback}\end{Bmatrix}$

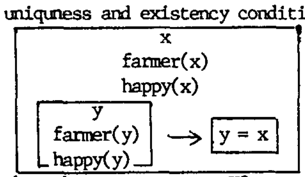
Therefore our selection set is restricted only by the accessibility relation and the descriptive content of the anaphoric NP's. Of course for anaphoric pronouns this content is reduced to a minimum, namely the grammatical features associated to them by the lexical entries. This accounts e.g. for the difference in acceptability of (10) and (11).
(10) Mary persuaded every man to shave himself
(11) *Mary promised 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 "himself" of (11) is replaced by x it cannot be identified with y having the (not explicitely shown) feature female.

$\begin{Bmatrix}(10')\\ (11')\end{Bmatrix}$



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 descrition holds for one referent only. Our algorithm proceeds as follows: First we build up a small IRS KO encoding the descriptive content of the common-noun-phrase of the definite description together with its uniquness and existency condition:

KO:



Second we have to show that we can prove KO out of the text-IRS of the preceeding 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 KO 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 explicitely in the discourse but follow implicitely 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 a 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 IRT contains Raising- and Equi-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 IRT to comprise such constructions. This is due to the fact that using LFG as syntactic base for IRT - and not the categorial syntax of Kamp - 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, Equi (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 Kamp'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 semantic from a semantic structure is associated. For the lexical entries of (3) we have

the following:

man —> man(*) a —> $\boxed{v}$

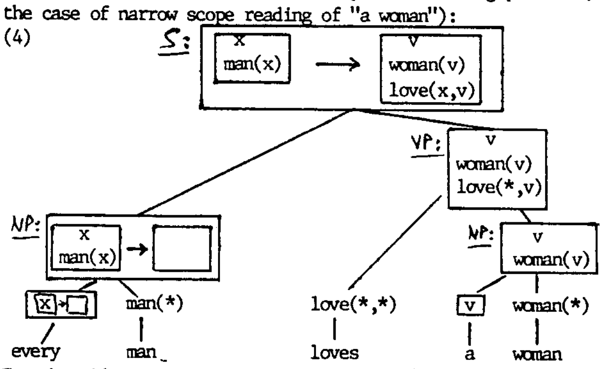
woman —> woman(*) every —> $\boxed{\boxed{x} \rightarrow \boxed{\ }}$

loves —> love(*,*)

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

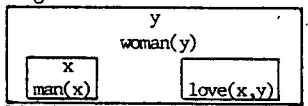
The DRS construction rules specify how these semantic structures are to be combined by propagating them up the tree. The easiest way to illustrate that is to do it by the following picture (for the case of narrow scope reading of "a woman"):

(4)



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

(5)



The picture should make clear the way we want to extend the parsing mechanism described in section 1 in order to produce DRS's as output and no more f-structures: instead of partially instantiated f-structures determined by the lexical entries partially instantiated DRS'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?

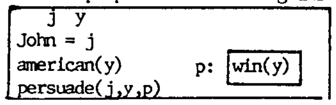
Recall that each clause has a unique head and that the functional features of each phrase are identified with those of its head. For (3) the head of S —> NP VP is the VP and the head of VP —> V NP is the V. Thus the outstanding role of the verb to determine and restrict the grammatical relations of the sentence is captured. (4) , however, shows that the logical 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 garantees that the logical head of a phrase is uniquely determied too. Consider

(6) John persuaded an american to win

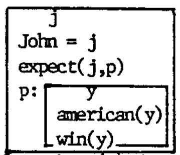
(7) John expected an american to win

for which we propose the following DRS's

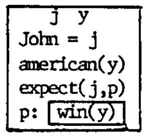
(6')



(7')

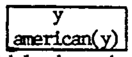


(7'')



The fact that (7) does not neccesserily 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 —> 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)

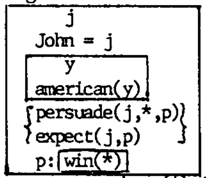


and thus would lead to the readings represented in (6') and (7''). Consequently (7') would not be produced.

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 mere n-place predicate but a DRS containing this predicate as atomic condition. This makes the unification possible but gives us the following result:

(*)



Of course one can say that (*) is open to produce the set of DRS's representing (6) and (7). But this means that one has to work on (*) 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. As the above example for the phrase VP —> 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 grammers 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/ Reyle, Uwe/ Rohrer, Christian, "Automatic Construction of a Knowledge Base by Analysing Texts in Natural Language", in: Proceedings of the Eigth Intern. Joint Conference on Artificial Intelligence II, 1983

[3] Halverson, P.-k., "Semantics for Lexical Functional Grammar". In: Linguistic Inquiry 14, 1982

[4] Kamp, Hans, "A Theory of Truth and Semantic Representa= tion". In: J.A. Groenendijk, T.U.V. (ed.), Formal Semantics in the Study of Natural Language I, 1981