The Treatment of Movement-Rules in a LFG-Parser

Hans-Ulrich Block, Hans Haugeneder
Siemens AG, München
ZT ZTI INF
W. Germany

In this paper we propose a way of how to treat long-distance movement phenomena as exemplified in 1) in the framework of an LFG-based parser.

(1) Who do you think Peter tried to meet
'You think Peter tried to meet who'

We therefore concentrate first on the theoretical status of so called wh- or long-distance-movement in Lexical Functional Grammar (LFG) and in the Theory of Government & Binding (GB), arguing that a general mechanism that is compatible with both LFG and GB treatment of long-distance-movement can be found. Finally we present the implementation of such a movement mechanism in a LFG-Parser.

The basic principles of the treatment of long distance phenomena or constituent control in LFG as described in (Kaplan/Bresnan 1982) can be characterized roughly in the following way:

1. The contextfree grammar is augmented by productions expanding to the empty word:

   $XP \rightarrow e$

   By the means of these productions traces are introduced at the "originating" position of a "displaced" constituent or speaking more in LFG terms a phonetically empty constituent is introduced at a c-structure position (the controllee constituent) whose role in the f-structure is to be played by the lexically specified linearly and structurally displaced constituent (the controller constituent).

2. Besides the immediate domination metavariables ($\uparrow, \downarrow$) another type of metavariables is used, the bounded domination metavariable, the double-ups and double-downs here represented $\uparrow\uparrow$ as and $\downarrow\downarrow$ for convenience. In contrast to the immediate domination variables, which allow to express identities of f-structures or f-structure parts assigned to c-structure nodes standing in the relation of immediate domination, the bounded domination metavariables allow to identify the f-structures of two c-structure nodes far apart in the c-structure.

The use of these metavariables can be seen in the following LFG grammar rules, where the symbol that introduces the controller constituent has among its equations one of the form $\downarrow = \downarrow\downarrow$ and the empty word is associated with the equation $\uparrow = \uparrow\uparrow$.

$$S' \rightarrow \underset{(\uparrow \text{ Q-FOC}) = \downarrow \quad \downarrow = \downarrow\downarrow}{NP} \quad \underset{\uparrow = \downarrow}{S}$$

$$NP \rightarrow \underset{\uparrow = \uparrow\uparrow}{e}$$

The instantiation procedure for these bounded domination variables thereby is defined such that it identifies the f-structures of the controller and controllee in a way which is shown in the following schematical description in figure 1 for a sentence like (2), leading to f-structures with shared substructures.

(2) [S' [NP *who*][S *does he try to find* [NP e]]S]S'

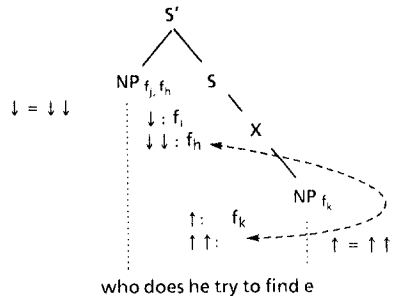

Figure 1

In addition to the basic approach presented so far there are some more means for expressing the various c-structurally defined constraints that have to be fullfilled in order to be allowed to identify two constituents' f-structures in the way just described and thereby rule out a lot of otherwise possible but ungrammatical structures.

The central additional mechanism to deal with these restrictions governing the possibliliy of establishing the constituent-control relation between two constituents is the use of bounding nodes. In LFG these are not specified globally but introduced at a rule specific basis, allowing for a natural treatment of the various idiosyncratic constituent control constructions in various languages that withstand a grammatical description by means of global bounding categories (see Kaplan/Bresnan (1982:245f) for a deeper discussion!).

Thus for example in the following VP- and S'-expansion introducing a sententiell wh-complement, the marking of S as bounding node accounts for the ungrammaticality of (3).

VP → (NP) S'

S' → whether **S**

(3) **What does he*[VP *wonder* [S' *whether* [S *she wants* [NP e]]S']S]VP

The blocking of structures like these is achieved by permitting no bounding node to lie on the path between the empty constituent and the root node of the control domain, which is specified as annotation of the $\downarrow\downarrow$-meatvariable ($\downarrow\downarrow_r$), shown in the following

rule.

$$S' \rightarrow \underset{(\uparrow \text{ Q-FOC}) = \downarrow \quad \downarrow = \downarrow \downarrow_S}{NP} \qquad \underset{\uparrow = \downarrow}{S}$$

With the fragmentary grammar rules just presented the identification of "what" with "e" in the embedded clauses object position is prevented as shown in Figure 2.

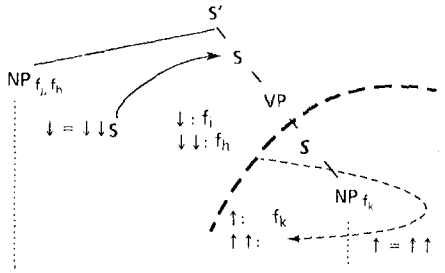

Figure 2

Here the occurrence of the bounding node (S) in the tree structure dominated by the root node of the constituent control domain effectively makes the S-dominated subtree inaccessible to any constituent control from outside.

[illegible] to constituent control (as categorial subscripts and additional feature restrictions) are not described as we do not treat them in our implementation.

In the GB-framework wh-Movement is regarded as an instance of the general transformation "move α". The wh-Phrase is moved out of its position (a.) into the COMP-Position dominated by S' and thereby leaves a trace in its original D-structure position (b.).

(4) a. $[_{S'}\_\_ [_S$ *have you done what*$_i]_S]_{S'}$
b. $[_{S'}$ *what*$_i$ $[_S$ *have you done* $t_i]_S]_{S'}$

wh-Movement underlies the constraint of subjacency but may be applied cyclically via the COMP-position, leaving traces in any COMP-position it meets:

(5) $[_{S'}$ *who*$_i$ $[_S$ *do you think* $[_{S'}$ t_i $[_S$ *Peter wants* $[_{S'}$ t_i $[_S$ PRO *to meet* $t_i]_S]_{S'}]_S]_{S'}]_S]_{S'}$

This assumption explains the ungrammaticality of (6) by the mere fact that the possible intermediate landing site for *what* is occupied by *who*.

(6) *$[_{S'}$ *What*$_i$ $[_S$ *do you think* $[_{S'}$ *who*$_j$ $[_S$ t_j *did* $t_i]_S]_{S'}]_S]_{S'}$

Furthermore this assumption explains the ungrammaticality of (7a) by the absence of a COMP-position in NP and the assumption that NP is a bounding node.

(7) a. *$[_{S'}$ *Who*$_i$ $[_S$ *do you believe* $[_{NP}$ *the claim* $[_{S'}$ t_i *that* $[_S$ *John loves* $t_i]_S]_{S'}]_{NP}]_S]_{S'}$
b. $[_{S'}$ *Who*$_i$ *do you think that* $[_S$ *John loves* $t_i]_S]_{S'}$

From a technical point of view, leaving aside philosophical and psychological reflections on universal grammar and learnability, the main difference between the two theories regarding wh-Movement can be seen in the following.

1) Whereas in LFG the application of wh-Movement is constrained by bounding nodes in GB it is constrained by bounding categories. GB is more restrictive in this point and aims towards a greater generalization, as it is excluded for two nodes with the same category to have different bounding node characteristics.

2) In GB the principle of subjacency which roughly says that no movement may cross two or more bounding nodes is used to explain the difference in grammaticality of (7a) and (7b) above. In LFG the same ungrammaticality is explained by the identity of the root node and the bounding node. The root node - being by definition a sister of the moved phrase - defines the subtree in which the trace for the moved phrase can and must be found. The fact that this root node may be crossed even if it is a bounding node but that the search is blocked if another bounding node is encountered evoques the subjacency effect.

3) The grammaticality of (5) is explained in GB by cylic COMP-to-COMP-movement. The NP in question is first moved into the first COMP-positon without violation of subjacency. Form there it is moved - again without violation of subjacency - to the next COMP-position. In LFG these facts are described by an additional grammar rule. The two rules for S' Kaplan/Bresnan (1982:241,253) suggests are:

$$S' \rightarrow \underset{(\uparrow \text{ Q-FOC}) = \downarrow \quad \downarrow = \downarrow \downarrow}{NP} \qquad \underset{\uparrow = \downarrow}{S}$$

$$S' \rightarrow (\textit{that}) \quad \underset{\uparrow \uparrow = \downarrow \downarrow}{\overset{\uparrow = \downarrow}{S}}$$

4) A last difference can be seen in the different grammar-modules long-distance-movement is treated in. In GB the movement is treated in a strictly constitutional way. It is part of the grammar's movement-rule-module and operates on nothing but positions. In LFG, long-distance movement is associated with the c-structure part of the grammar as well as with the part that is repsonsible for the instantiation of the variables, especially the bounded domination metavariable. Thus - although being introduced via phrases in the c-structure rules - it effectively does not operate on phrases but on functional variables.
(According to Ron Kaplan (personal communication) a totally f-structure-oriented approach of such phenomena will be given in the ongoing development of LFG, whose exact elaboration is forthcoming)

When constructing our parser we found it very enlighting to "merge" the two theories in the following way. We

- consider subjacency as an epiphenomenon of the fact that 1) bounding nodes are strict boundaries for movement and 2) a bounding node may be

crossed if it is a root node (as in LFG).

- adapt the more flexible LFG view of (rule-specific) bounding nodes instead of bounding categories, as one can anyway derive bounding nodes from bounding categories by replacing every occurrence in the grammar rules' right hand side of a bounding category by the bounding-node-marker.

- prefer the principle of cyclicity over equations of the type $\uparrow\uparrow = \downarrow\downarrow$.

- treat movement strictly constitutionally.

Our parser is based on Earley's (1970) algorithm augmented by a mechanism for treating long-distance phenomena according to the above mentioned principles. The basic context-free parser operates on two ordered sets of states that correspond to the state sets s_i and s_{i+1} in the Earley-Parser to the end of which constantly new states which are still to be worked on are added. A state is a quintupel (<tree> <left> <right> <dot> <pred.-list>).

- <tree> is the current parsetree of that path,

- <left> and <right> are pointers to the input string the constituent begins with and the input string that immedeately follows the constituent respectively,

- <dot> marks the current position in the right side of the cf grammar rule and

- <pred.-list> is a set of pointers to all preceeding states who's treenodes might become the mother of current states' tree.

A treenode is a complex data structure that contains the node's label (i.e. its syntactic category), a list of its daughters and a pointer to the f-structure attached to it.

The basic operations are *predict*, *scan* and *complete* which are close to the definition in Earley (1970). For the construction of the c-structure these actions are augmented in the following way: *predict* creates an empty treenode labeled with the predicted category, *scan* attaches the next input word as the rightmost daughter to the state's <tree>, and *complete* attaches the state's <tree> as the rightmost daughter to all treenodes in the states of the current state's <pred.-list>.

For the construction of the f-structure, which is built up incrementally being used as a filter on c-structures as soon as possible (as described in Block/Hunze(1986)) the following augmentations are performed: The <dot> part of a state not only marks the position in the cf-rule's right hand side but also contains the functional equations associated with that position. When *completing* and *scaning* the parser instantiates the up- and down-arrow of the equations with virtual copies of the mother's and daughter's f-structure. The equations are then evaluated and the new f-structure associated with the up-arrow becomes the f-structure of the new state's tree.

The basic idea behind the treatment of long-distance phenomena is to augment the mechanism for the contextfree skeleton of the grammar with a mechanism that transports displaced elements until they can be consumed at suitable postions. The suitability of these positions thereby is restricted by several constraints. Firstly a position only can come into account if a phrase of the same type as the moved phrase is predicted by the grammar. Secondly the mechanism for propagating the moved constituents obeys certain linguistic constraints such as bounding nodes. Thus it roughly can be viewed as a sort of linguistically constrained HOLD/VIR-mechanism that is integrated in the parser, freeing the grammar writer from the necessity of encoding the details of holding displaced elements and consuming them by VIR explicitly on the grammar level.

An advantage of this approach for the handling of such phenonena is that one can do without empty productions in the cf part of grammar that tend to lead to an enormous amount of spurious phrase structure trees. On the other hand the mechanism to be presented shows some asymetry, since it only deals with long distance dependencies where the displaced element occurs in terms of parsing direction before its originating position. Since the parser presented here parses form left to right this means that we take the assumption for granted that there are no rightward long-distance movements.

For the treatment of movement <dot> is expanded to the complex data-structure of the type (<syn. cat.> <equations> <slash>) where <slash> is a flag containing information on constituents to be moved. If <slash> is empty no movement is performed. If <slash> is set the node which is associated to the <dot> will be moved rightward.

A state is augmented by three additional components, namely <pending> and <consumed> <to-be-moved> that are used for the bookkeeping of moved nodes.

<pending> is a pushdown stack of the nodes that are moved. Each time a node is declared in the grammar to be moved it is pushed onto the <pending> of the state being developed. The nodes in <pending> are then propagated to the subsequent paths.
<consumed> is the list of all traces consumed in a subtree. That is whenever a constituent on <pending> is used to satisfy the corresponding prediction of some state (i.e. being used as if it was the current element in the input at some state) it is popped from <pending> and pushed on <consumed>. Furthermore <consumed> is used to control the attachment of phrases to their mother nodes by the completer, allowing phrases dominated by a root node with consumed subphrases in them to be attached only if that phrase is also in <pending> in the mother state.

<to-be-moved> is used to transport a displaced constituent to its corresponding root-node, where after being pushed onto <pending> it can be consumed as a missing constituent

The snapshot of the parser's states in (9) shows the relevant subset of states induced by the attachment of an wh-NP by a grammar rule like (10) whilst parsing a sentence like the one in (8).

(8) (I don't know) who he loves?
1 2 3 4

(9)

<tr><l,r><dot><plst><pg> <cd><tbm>

[S'] (1,1) .NP .. NIL NIL NIL (1)

..

[NP_{who}]

(1,2) EOR (1) NIL NIL NIL (2)

[S'[NP_{who}]]

(1,2) .S NIL NIL NIL NP_{who} (3)

(the parsed wh-NP "who" is moved to its root-node)*

[S] (2,2) .NP (3) (NP_{who}) NIL NIL (4)

(in order to be available for consumation it is pushed on <pending>)*

..

[S[NP_{he}]]

(2,3) .VP (3) (NP_{who}) NIL NIL (5)

..

[VP[V_{love}]]

(2,4) .NP (5) (NP_{who}) NIL NIL (6)

[NP] (4,4) .PN (6) (NP_{who}) NIL NIL (7)

[VP[V_{love}][NP_{who}]]

(3,4) EOR (5) NIL (NP_{who}) NIL (8)

(the moved NP is consumed as direct object of "love")*

..

[S[NP_{he}][VP[V_{love}][NP_{who}]]]

(2,4) EOR (3) NIL (NP_{who}) NIL (9)

[S'[NP_{who}]][S[NP_{he}][VP[V_{love}][NP_{who}]]]]

(1,4) EOR NIL NIL NIL NIL (10)

(after completing a state with a root-node category predicted, the moved NP is taken out of <consumed>, meaning that the structure, where it has to be consumed is closed and the movement is performed completely)*

(10) S' → NP ($\uparrow$ Q-FOC) = $\downarrow$ $\downarrow$ = $\downarrow\downarrow$ $\quad$ $\underline{S}$ $\uparrow = \downarrow$

(with S being a root-node)

In order to see the working of our mechanism in the case of a cyclic movement a look at the following examples shows its basic features. Thus in an example like

(11) Who do you believe that he knows
1 2 3 4

the moved wh-NP is transported into the embedded sentence via landing at a S'-initial optional NP-position, from where it is moved in turn.

(12) S' → (NP/$\uparrow$) $\underline{S}$

(with S being a root-node again)

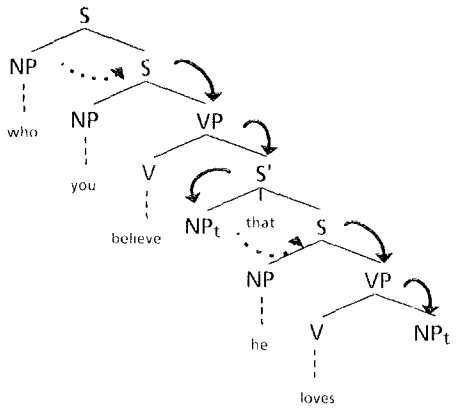

Figure 3

With a grammar rule like (12) - where XP/$\uparrow$ in our rule notation means that the phrase XP is to be moved - the wh-NP who is first moved form its matrix-sentence initial position and consumed at the embedded sentences' NP-position from where it is moved again as shown in Figure 3.

The treatment of the three critical components <pending>, <consumed> and <to-be-moved> in such a case of cyclic movement is shown in the following partial trace

(13)

<tr><l,r><dot><plst><pg> <cd><tbm>

[VP[$V_{believe}$]]

(1,1) .S' .. (NP_{who}) NIL NIL (1)

(when the embedded sentence is predicted the moved constituent is already pending)*

[S'] (1,1) .NP$\uparrow$ (1) (NP_{who}) NIL NIL (2)

..

[S'[NP_{who}]]

(1,1) .COMP NIL NIL (NP_{who}) NP_{who} (3)

(the moved constituent is consumed and moved in turn by putting it on <to-be-moved>)*

[S'[NP_{who}]$_{that}$]

(1,2) .S (1) NIL (NP_{who}) NP_{who} (4)

[S] (2,2) .NP (4) (NP_{who}) NIL NIL (5)

(in order to be available for consumation it is pushed on <pending>)*

..

[VP[V_{love}][NP_{who}]]

(3,4) EOR (5) NIL (NP_{who}) NIL (6)

(the moved NP is consumed as direct object of "love")*

[S[NP_{john}][VP[V_{love}][NP_{who}]]]

(2,4) EOR (3) NIL (NP_{who}) NIL (7)

[S'[NP_{who}]$_{that}$][S[NP_{john}][VP[V_{love}][NP_{who}]]]]

(1,4) EOR (1) NIL (NP_{who}) NIL (8)

[VP[$V_{believe}$][S'[NP_{who}]$_{that}$][S[NP_{john}][VP[V_{love}][NP_{who}]]]]]

(..,4) EOR .. NIL (NP_{who}) NIL (9)

(the moved wh-NP is still on <consumed>, waiting to be discarded when the state predicting its originating root-node category is completed)*

The organisation of the list of <pending> nodes as a pushdown-stack rather than a queue mirrors the property of long-distance movement to be nested. The parser will therefore account for the ungrammatical trace bindings in (14).

(14) *Which sonata$_i$ is this violin$_j$ easy to play t_i on t_j

Though the sentence will finally be parsed, it will have, as predicted, the semantically deviant nested binding of the traces: *to play the violin on which sonata.*

The mechanism presented so far does not cover the treatment of bounding nodes, as there are no bounding restrictions on the way how the constituents on <pending> are transported and/or consumed. Without imposing any further restrictions on our mechanism, it is possible to move a constituent into a subtree dominated by a bounding node.

To prevent this a new empty <pending> is used in every state that is a consequence of the prediction of a bounding node category. Thus any moved constituent which is possibly on <pending> at such a state of the analysis is not available during the parsing of the bounded node category dominated substructure. When, however, this substructure is parsed completely and attached to its mother structure by the completer the old (i.e. the mother's state) <pending> is used and propagated in the subsequent states.

Thus in an example like (15) if at some state the bounding node category S (***S***) is predicted all the successive states (as for example state$_j$) have a new empty <pending>.

(15) **What does he*[$_{VP}$ *wonder* [$_{S'}$ *whether* [$_{S}$ *she wants* [$_{NP}$ e]]$_S$]$_{S'}$]$_{VP}$

(16) .....whether she wants
1 2 3 4

<tr><l,r><dot><plst><pg> <cd><tbm>

[S' [V wonder]]

(1,1) .***S*** .. (NP_{who}) NIL NIL (i)

(when the bounding node category is predicted "what" is pending; it will not be propagated to those subsequent states that expand the bounding node doinated substructure)*

[S] (1,1) .NP .. NIL NIL NIL (j)

[S she wants]

(2,4) .EOR .. NIL NIL NIL (k)

[S' whether][S she wants]]

(1,4) .EOR (i) (NP_{who}) NIL NIL (l)

(after the completion of the **S**-structure the old <pending> is activated again, making "what" acessible again)*

[S' what .. ..wants]

(..,4) .EOR NIL (NP_{who}) NIL NIL (m)

In our example the parser will come to state l after completing state i and finally (via some more completer operations) to state m. State m finally represents a configuration that says that (14) will not be parsed due to the fact that <pending> is not empty while the complete input is analysed with no predictions left.

For the specification of bounding nodes in our grammar we offer three possibilities. Firstly in the spirit of LFG on a rule specific basis, secondly globally by declaring a category a globally bounding node (which diminishes the grammar writers work on actually globally bounding nodes) and thirdly a negative specification concerning bounding features of a globally bounding node, thus admitting an simple expression of exceptions.

The parser pesented is implemented in Interlisp-D on a personal Lisp workstation and has been testet with a grammar comprising a major part of the phenomena discussed in Kaplan/Bresnan's fundamental LFG-paper.

The work described here is partly sponsored by the Federal Ministry of Research and Technology in the WISBER-project.

We would like to express our thank to our collegues M. Gehrke and R. Hunze for their criticism and encouragement.

Literature

Block, H.-U. and Hunze, R. (1986)
Incremental Construction of C- and F-Structure in an LFG-Parser, Proc. COLING-86 (to appear)

Earley, Jay (1970)
An Efficient contextfree parsing algorithm. *Communications of the ACM* 6(8), 541-455

Kaplan, Ronald M. and Bresnan, Joan (1982)
Lexical-Functional Grammar: A Formal System for Grammatical Representation in: Bresnan, Joan (ed): *The Mental Representation of Grammatical Relations.* Cambridge/Mass.