A Concept of Derivation for LFG

Jürgen Wedekind
Department of Linguistics
University of Stuttgart
West Germany

Abstract
In this paper a version of LFG will be developed, which has only one level of representation and is equivalent to the modified version of [2], presented in [3]. The structures of this monostratal version are f-structures, augmented by additional information about the derived symbols and their linear order. For these structures it is possible to define an adequate concept of direct derivability by which the derivation process becomes more efficient, as the f-description solution algorithm is directly simulated during the derivation of these structures, instead of being postponed. Apart from this it follows from this reducability that LFG as a theory in its present form does not make use of the c-structure information that goes beyond the mere linear order of the derived symbols.

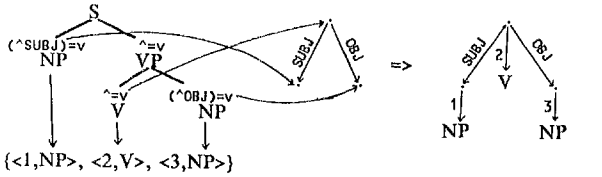
1.Introduction*
The derivation process of sentences in LFG as defined in [2] is very complex, because an additional filter has to be applied to derived c-structures. Within this filter component the f-structures are constructed. An f-structure can be regarded as a special kind of labelled directed acyclic graph (DAG), which represents a structure of partial functions (i.e. the labels of the edges leaving each node have to be different). 1) A terminal string (x) is wellformed if it satisfies the following conditions:

i. There is a c-structure for x that can be generated by the contextfree (cf) base of the grammar.
ii. There is an f-structure d and a mapping f from the set of nodes of c to the set of subDAGs of d such that d is a unique minimal f-structure that satisfies the annotations associated with the c-structure nodes (f and d are constructed by the f-description solution algorithm, for short KB-algorithm).
iii. d satisfies all constraints in the f-description.
iv. d is complete and coherent.

The derivation process is performed by checking these conditions in the order i. < ii. < iii.,iv..
This kind of derivation has several theoretical and practical disadvantages:

a. A parser/generator that works in accordance with this derivational process is not very efficient, because usually many strings are parsed completely or many c-structures are generated completely, although the strings themselves are not wellformed and therefore rejected by the filter (ii.-iv.).
b. The process is formally not very transparent and complicates a comparison with other formalisms.
c. The grammar is multistratal. It has two levels of representation (c-structures, f-structures), although it can be shown that the c-structure information, which goes beyond the linear order of the derived symbols is not exploited in the present version of the theory ([2],[3]) at all. 2)

In the following it will be demonstrated that for each LFG a monostratal version can be constructed, which describes the same language with the monostratal concept of derivation. The entities of these derivations are augmented f-structures which are constructed along the following lines: In the KB version the c-structure derivation of a wellformed terminal string is a sequence of annotated c-structures, where each c-structure results from the application of an annotated cf rule to a terminal node of the preceding c-structure. If one applies the KB-algorithm to each annotated c-structure then it constructs a minimal f-structure and a mapping from the c-structure nodes to the substructures of the f-structure. The augmented f-structures are then constructed by attaching the occurrences of the derived string, represented by the labelled terminal nodes of the c-structure tree, as additional labelled edges to the start nodes of those subDAGs, which are the values of the mapping for the corresponding c-structure nodes. An example:



The reduction will be complete, if a definition of an adequate concept of direct derivability for these structures can be given. This will be done here in three steps in the following sections. The reducability to these structures will show that the LFG makes no essential use of c-structure informations, as the new structures contain only information about the linear order of the derived symbols and no other overt or hidden c-structure information. (The wellformedness conditions iii. and iv. are defined on f-structures.) The reduction leads to a more efficient derivation process, because the postponed filter component ii. (the KB-algorithm) is integrated in the concept of direct derivability. 3)

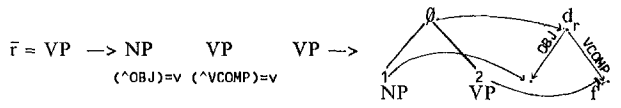
2. Derivation of f-description solutions parallel to c-structures
In this section the derivation process of KB will be modified in such a way, that in each step, besides a partial c-structure, also that partial f-structure is generated, which would be the result of the KB-algorithm, if it were applied to the annotated partial c-structure. A derivation is then a special sequence of triples consisting of a partial c-structure, a partial f-structure and a mapping from the c-structure nodes to the subDAGs of the f-structure. Before stating the definition for the start entity and the concept of direct derivability the derivation concept will be defined analogously to the standard derivation concepts.

DEF A derivation is a sequence $s_0 \ldots s_n$ where
$s_i =: <c,d,f>$; $0 \leq i \leq n$; c =: c-structure; d =: partial f-structure
$f \in [N(c) \rightarrow T(d)]$ (mapping from the c-structure nodes (N(c)) to the subDAGs of d (T(d)))
$s_0 =:$ start triple
$s_n =:$ derived triple; with a c-structure of a terminal string
$s_{i-1} \overset{r}{\Rightarrow} s_i$; $0<i\leq n$ (s_i follows from s_{i-1} by rule r)

The c-structure of the start triple consists, as usual in c-structure derivations, only of the start node (1) with the label S ({<1,S>}). If each node is to be in the domain of f, each node (including the start node) has to be annotated with a 'v' metavariable. Thus it is possible to apply the KB-algorithm to the start node. The algorithm creates a place-holder (DAG, consisting only of one node), to which the c-structure node is mapped. So it is adequate to introduce for d_0 a minimal DAG to which the c-structure node (1) is mapped qua f^0.

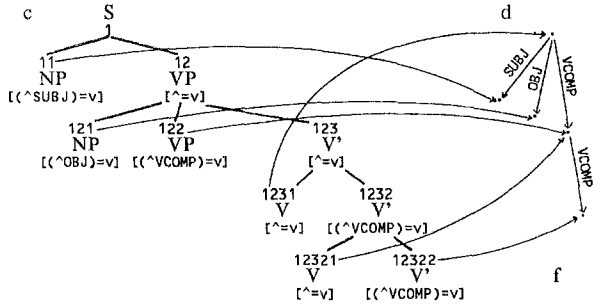
$s_0 =: <\{<1,S>\}, d_0, f^0>$ $\qquad f^0$ $\qquad$ S $\overset{}{\longrightarrow}$ d_0 (1)

As the entities of a derivation are complex, an application of a rule to a triple expands each component of that triple. The expansion, that is achieved by a rule, can then be isolated if one applies the KB-algorithm to the annotated c-structure which is introduced by this annotated rule. Example:

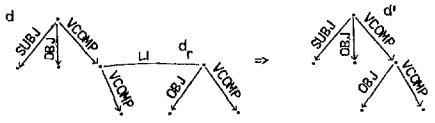


Thus it is possible to construct for each annotated cf rule $\bar{r}$ with cf base r a rule of the form $<p_1(r),<\{<\emptyset,p_2(r)>\},d_r,f^r>>$ where $p_1(r)$ is the first projection of r (the lefthand side of the rule r which equals the lefthand side of $\bar{r}$) and $\{<\emptyset,p_2(r)>\}$ is the introduced c-structure. (If one represents the cf part of the righthand side of the annotated rule $\bar{r}$ $(p_2(r))$ as a sequence of symbols in the set theoretical sense, it is possible to interpret the integers as nodes and the symbols as labels of the nodes.
For the above rule: $\{<\emptyset,\{<1,NP>,<2,VP>\}>\}$.)

The concept of direct derivability now has to be defined as follows: The application of a constructed rule to a triple, which is constructed for an annotated c-structure by the KB-algorithm, has to yield exactly that expansion that would be the result of the KB-algorithm, if it were applied to the annotated c-structure which is derived by the corresponding annotated rule. Before stating the expansions formally we will describe them informally and illustrate them by an example. That the definition of direct derivability is adequate in this sense can only be sketched here. Assuming that the triple $s_{i-1}=<c,d,f>$



is derived and the rule $\bar{r}$ (above) is applied to node 122, the new components of $s_i=<c',d',f'>$ are determined as follows: 4)
1. (c-structure) The new c-structure is the expansion which results from applying r to a terminal node (122 in the example) which is labelled with the lefthand symbol of the rule.
2. (f-structure) As the c-structure, introduced by the rule, expands the node 122 in c', all '^' metavariables in the annotations of the rule have to be instantiated by 122 (and not by $\emptyset$). Thus $'f_{122}=f^r_\emptyset$' and it is necessary to merge the subDAG of d denoted by dVCOMP with d_r. 5) The new DAG d' is then the minimal extension of d which results from unifying the DAG which is introduced by the rule with that subDAG of d to which the expanded node was mapped qua f.

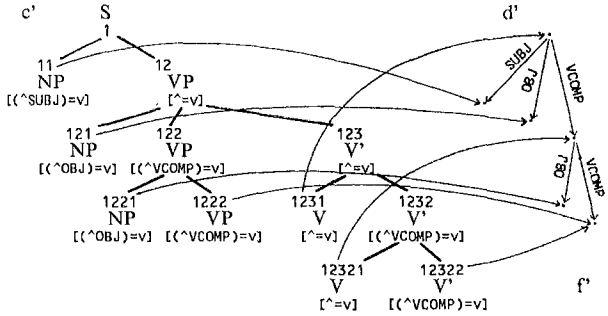


3.(mapping) As the new DAG is an extension of d all attribute paths of d are also paths of d'.
a) If the value of f for a node $\in$Dom(f) is denoted in d by dp, it's f' value in d' will be denoted by d'p.
b) For the new nodes f' is defined as follows:
As the c-structure introduced by the rule expands 122 in c', the node $\emptyset$ is identified with 122. Therefore $'f_{122}=f^r_\emptyset$' and by the application of the Merge operator dVCOMP⊔d_r is constructed as the (new) substructure of d', denoted by d'VCOMP. By the application of the Substitute operator d'VCOMP becomes the (new) value of f'_{122}. By the expansion of 122, a node j of the c-structure, introduced by the rule, becomes the node 122j in c'. If $'f'_j=d_rp'$' then by the recursion of Merge dVCOMPp'⊔d_rp' is constructed as the (new) substructure of d', denoted by d'VCOMPp'. By the application of Substitute dVCOMPp'⊔d_rp' becomes the (new) value of f'_{122j}. Therefore: If the value of f^r for the node j ($\emptyset$<j) of the rule's c-structure is denoted by d_rp' and the value of f for the expanded node i is denoted by dp, then the value of f' for ij is denoted by d'pp'. For the new nodes in the example the values of f' are determined as follows:

$f^r_1=d_r$OBJ, $f_{122}=$ dVCOMP => $f'_{1221}=$ d'VCOMP OBJ

$f^r_2=d_r$VCOMP, f_{122} dVCOMP => $f'_{1222}=$ d'VCOMP VCOMP

and <c',d',f'> is



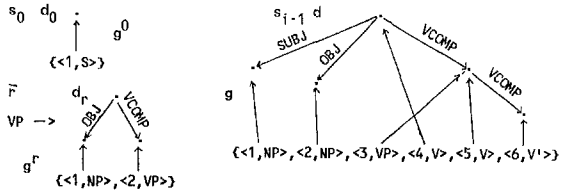
The concept of direct derivability is defined in the following way:

DEF $<c,d,f> = s_{i-1} \xrightarrow{\bar{r}} s_i = <c',d',f'>$ <->
$c \xrightarrow{r} c'$ and if node i is expanded

$d'=\sqcap\{d''\in DAG \mid d\sqsubseteq d'' \wedge \forall p\in PATH(f_i=dp \rightarrow d''p = dpud_r)\}$
$f'\in [N(c') \rightarrow T(d')]$
$\forall j\in Dom(f)\forall p\in PATH(f_j=dp \rightarrow f'_j=d'p)$
$\forall j\in Dom(p_2(r))\forall p,p'\in PATH(f_i=dp \wedge f_j\sqsubseteq d_rp' \rightarrow f'_{ij}=d'pp')$

3. Derivation of f-description solutions parallel to strings

Since in the derivation process sketched above only the values of f for the terminal nodes are needed to define the next triple, it is possible to define an alternative version for strings instead of c-structures. The c-structure information which goes beyond the linear order of the labelled terminal nodes is not required. Derivations are to be defined for tripels <w,d,g> in analogy to Chap. 2, with w being a string, d a DAG and g$\in$[w' -> T(d)]. The entities of a derivation and the triples which are introduced by the rules can be constructed from the entities of the preceding version easily:
If <c,d,f> is such an entity of the preceding version then w is the final string represented by c, the DAG equals d and g maps each occurrence of the final string represented by c to that subDAG of d which is the value of f for the corresponding terminal node. The start triple is $s_0=<\{<1,S>\},d_0,f^0>$ and a rule has the form $<p_1(r),<p_2(r),d_r,g^r>>$. By this construction one obtains the following entities for s_0 and for the examples $\bar{r}$ and s_{i-1} of the preceding section:

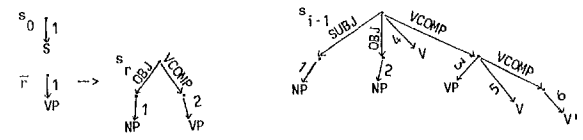


The definition of an adequate concept of direct derivability is now relatively simple. Assuming that $s_{i-1}=<w,d,g>$ and $s_i=<w',d',g'>$:
1.(string) The direct derivability for the strings is defined as usual:
w' follows from w iff there is a rule $\bar{r}$ and w' follows by the cf part of $\bar{r}$ from w.

2.(f-structure) Because
a.the DAG, to which the expanded node is mapped in the c-structure version equals that DAG (dp), to which the replaced occurrence is mapped and
b.the DAG d_r, which is introduced by a rule, is in both versions the same; it follows that the minimal extension of d, which results from $dp \sqcup d_r$, equals the derived DAG in the c-structure version.
3.(mapping) To define g' we have to account for:
a.The indices of the occurrences of w corresponding to the terminal nodes in the right context of the expanded node in the c-structure version are increased at the transition to w' depending on the length of the righthand side of the applied rule ($|p_2(r)|$).
b.The indices of the occurrences of $p_2(r)$ which are used to construct the new nodes in the c-structure version are here increased depending on the index of the replaced occurrence. This leads to the following definition:

DEF $<w,d,g> = s_{i-1} \xrightarrow{\bar{r}} s_i = <w',d',g'> <->$
$w \xrightarrow{\bar{r}} w'$ and if $<i,w_i>$ is the replaced occurrence
$d'=\sqcap\{d'' \in DAG | d \sqsubseteq d'' \wedge \forall p \in PATH(g(<i,w_i>)=dp \rightarrow d''p=dp \sqcup d_r)\}$
$g' \in [w' \rightarrow T(d')]$
$\forall j \in Dom(w')(j<i \rightarrow \forall p \in PATH(g(<j,w_j>)=dp \rightarrow g'(<j,w_j'>)=d'p))$
$\forall j \in Dom(w')(j \geq i+(|p_2(r)|) \rightarrow \forall p \in PATH(g(<j-(|p_2(r)|-1)w_j'>)=dp \rightarrow g'(j,w_j')=d'p))$
$\forall j \in Dom(w')(i \leq j < i+|p_2(r)| \rightarrow \forall p,p' \in PATH(g(<i,w_i>)=dp \wedge g^r(<j-(i-1),w_j'>=d_r p' \rightarrow g'(<j,w_j'>)=d'pp'))$

4. The monostratal derivation concept
One obtains the monostratal version quite naturally if for all triples <w,d,g> of a derivation and a rule's right side in the string version the arguments of g are attached as additional labelled edges to the start nodes of the subDAGs of d to which they are assigned qua g. Thus one obtains for the start triple, the derived triple s_{i-1} and the rule $\bar{r}$ the following structures: 6)



If <w,d,g> is a triple of the string version then a DAG

$s = \sqcap\{d' \in DAG | d \sqsubseteq d' \wedge \forall <i,w_i> \in w \ \forall p \in PATH(g(<i,w_i>)=dp \rightarrow d'p(i)=w_i)\}$

is the corresponding entity in the monostratal version. These entities are augmented partial f-structures. They have additional terminal edges. These edges are labelled with integers and lead to elements of the vocabulary. The labels of such edges which are attached to different nodes have distinct labels. All such edges represent a string over V.
If FSp is the (undefined) set of partial f-structures then the new structures are elements of the set FSe

$FSe=\{s \in DAG | \exists d \in FSp \exists w \in V^* \exists g \in [w \rightarrow T(d)](s=\sqcap\{d' \in DAG | d \sqsubseteq d' \wedge \forall <i,w_i> \in w \forall p \in PATH(g(<i,w_i>)=dp \rightarrow d'p(i)=w_i))\}$

The string of such a structure s (S(s)) is simply the set of all these additional edges.

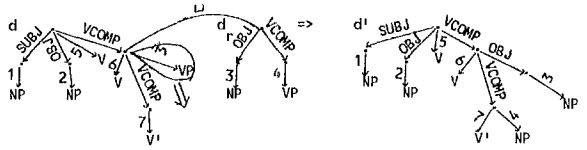
DEF $\forall s \in FSe(S(s)=\{<i,x> \in \mathbb{N} \times V | \exists p \in PATH(sp(i)=x)\}$

The derivation concept is defined as follows:

DEF A derivation is a sequence $s_0 \ldots s_n$ where

$s_i \in FSe$; $0 \leq i \leq n$; $s_0 =: \xrightarrow{1} S$
$S(s_n) \in V_T^*$; $s_{i-1} \xrightarrow{\bar{r}} s_i$; $0 < i \leq n$

As in this version the occurrences are attached as edges to the start nodes of those subDAGs, to which they are assigned in the string version, an adequate concept of direct derivability can be inferred from the definition of the preceding section.
One properly re-indexes the DAG s_{i-1} and the DAG, which is introduced by the rule (d,d_r), according to the definition of g'.
The derived DAG is constructed by the elimination of the edge to be replaced, and the unification of d_r with that subDAG of d, to which the replaced edge was attached. A successful unification in the string version can't fail here because the labels of the additional edges of d (after elimination of the edge to be replaced) and d_r are pairwise different.
In the example the result of the application of $\bar{r}$ on <3,VP> of s_{i-1} is defined as follows:



The concept of direct derivability is defined in the following way:

DEF $s=s_{i-1} \xrightarrow{\bar{r}} s_i=s'$ <->
$\exists p \in PATH \exists i \in \mathbb{N}(sp(i)=p_1(\bar{r}) \wedge s'=\sqcap\{d' \in DAG | d \sqsubseteq d' \wedge d'p=dp \sqcup d_r)\}$
with C as the set of atomic values and s_r as righthand side of $\bar{r}$

$$d_r=\sqcap\{d' \in DAG \mid \begin{array}{l} \forall p \in PATH(s_r p \in C \rightarrow d'p=s_r p) \wedge \\ \forall p \in PATH(s_r p \backslash S(s_r)=\emptyset \rightarrow d'p \backslash S(d')=\emptyset) \wedge \\ \forall p \in PATH \forall j \in \mathbb{N}(s_r p(j)=d'p(j+(i-1))) \end{array}\}$$

$$d=\sqcap\{d' \in DAG \mid \begin{array}{l} \forall p \in PATH(sp \in C \rightarrow d'p=sp) \wedge \\ \forall p \in PATH(sp \backslash S(s)=\emptyset \rightarrow d'p \backslash S(d')=\emptyset) \wedge \\ \forall p \in PATH \forall j \in \mathbb{N}(j<i \rightarrow d'p(j)=sp(j)) \wedge \\ \forall p \in PATH \forall j \in \mathbb{N}(j>i \rightarrow d'p(j+|S(s_r)|-1)=sp(j)) \end{array}\}$$

FOOTNOTES:

* The material in this paper is based on work supported by the BMFT under grant no. 1013207 0.

1) [2] suggests, that atomic feature values are not represented as labelled nodes. Thus in the following illustrations only edges labelled with complex valued features (grammatical functions) lead to nodes; edges labelled with atomic valued features (morphological features) point at the atomic values.
2) In this version (cf. [3]) long distance dependencies are handled on f-structure level. For that purpose regular expressions over the set of nuclear functions (governable functions plus ADJ, XADJ) are allowed to occur in the equations. These rules can be interpreted as schemata. A rule which is an instance of a schema is then annotated with an expression that is element of the set, denoted by the regular expression.
3) This integration is necessary because f-structures are control structures of the filter component ii. and the new structures are expanded f-structures. It is also possible to simulate the postponed filter components iii. and iv. in an adequate way during the derivation. This can't be discussed here for lack of space.
4) This example is derivable with the grammar of [1]. The annotations are attached to the nodes in order to make d and f reconstructable. I represent nodes as sequences of integers in the usual way (start node 1; ij is the j-th daughter of the node i). For reasons of clarity f is specified only for the terminal nodes.
5) If d is a DAG and p a path (a sequence of attributes), then dp is an abbreviation of a term (descriptor) denoting a subDAG of d. The actual structure of such a term depends on the chosen metatheoretical reconstruction of DAGs (partial functions vs. graphs) (cf. eg. [4]).
6) Note that the VCOMP substructure comprises a discontinous structure whose corresponding symbols do not form a proper substring in w.

REFERENCES

[1] BRESNAN, J./R.KAPLAN/S.PETERS/A.ZAENEN (1982), Cross-Serial Dependencies in Dutch. In: Linguistic Inquiry 13.4
[2] KAPLAN, R./J.BRESNAN (1982), Lexical Functional Grammar. A Formal System for Grammatical Representation. In: BRESNAN, J. (ed.), The Mental Representation of Grammatical Relations. Cambridge, Mass. 1982
[3] KAPLAN, R/A.ZAENEN (1986), Functional Incertainty in LFG. unpubl. ms., Stanford
[4] PEREIRA, F.C.N./S.M.SHIEBER (1984), The Semantics of Grammar Formalisms Seen as Computer Languages. In: Proceedings of COLING 84