Type-Driven Semantic Interpretation of f-Structures

Jürgen Wedekind
Institute for Natural Language Processing
University of Stuttgart
Azenbergstr. 12
D-7000 Stuttgart 1, FRG
juergen@ims.uni-stuttgart.de

Ronald M. Kaplan
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304 USA
Kaplan.Parc@Xerox.Com

Abstract

The formal architecture of Lexical Functional Grammar offers a particular formal device, the structural correspondence, for modularizing the mapping between the surface forms of a language and representations of their underlying meanings. This approach works well when the structural discrepancies between form and meaning representations are finitely bounded, but there are some phenomena in natural language, e.g. adverbs in English, where this restriction does not hold. In this paper, we describe rule-based type-driven interpretation algorithms which cover cases of such a structural misalignment by exploiting a new descriptive device, the "restriction operator". The algorithms are set up in such a way that recursive rules can be derived for the interpretation of adjunct sets within a *codescription* approach (see [Kaplan and Wedekind, 1993] for details).

1 Introduction

In [Kaplan and Bresnan, 1982] Lexical Functional Grammar (LFG) was introduced as a grammatical formalism that assigns to a sentence entities of two different levels of representation: a c-structure representing information on the structure of the phrases of a sentence and an f-structure which represents its underlying predicate-argument structure. The structures are set in correspondence by a function from the c-structure nodes (constituents) into the substructures of the f-structure. The f-structure is identified with the smallest structure that satisfies the f-description, a description of the f-structure which is built up by instantiation of the annotations of the context-free rules and projected off the c-structure by the correspondence mapping.

This architecture was then extended by Kaplan [1987] and Halvorsen [1987] to structures representing information on other levels of linguistic representation. These structures (called projections) are *codescribed* by the annotations of the context-free grammar and set in correspondence by additional projectors. Furthermore, Kaplan *et al.* [1989] applied the general correspondence architecture to the problem of translation by projecting from the f-structure of a sentence of a given source language an additional f-structure of its translation into some target language.

Within the domain of semantic interpretation, which is the topic here, the semantic structures are the range of the σ-projector which maps substructures of the f-structure into corresponding substructures of the semantic structure. In figure 1, the

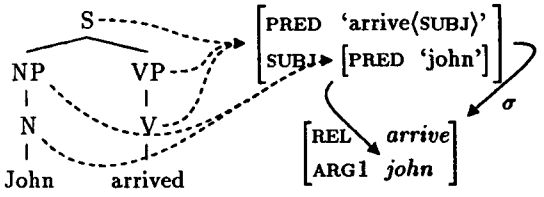

Figure 1
Structural correspondences between c-, f- and σ-structure.

semantic structure (σ-structure) and the structural correspondence between f- and σ-structure for the sentence *John arrived* are codescribed by additional

annotations of the lexical entry for *arrived* in (1).

(1) arrived: V, $(\uparrow \text{PRED}) = \text{'arrive}\langle\text{SUBJ}\rangle\text{'}$
$(\sigma \uparrow \text{REL}) = \textit{arrive}$
$(\sigma \uparrow \text{ARG1}) = \sigma(\uparrow \text{SUBJ})$

Within the domain of translation, Wedekind [1988], and Sadler and Thompson [1991] recognized some problems of the correspondence approach which concern data of *head-switching*. These difficulties also arise in the domain of semantic interpretation. In the latter domain we find constructions where the syntactic head (the predicate) does not correspond to the semantic head as, e.g., in adverbially-modified sentences like (2)

(2) John arrived late

whose f- and σ-structure are given in figure 2. In

$$f: \begin{bmatrix} \text{ADJ} & \{a:[\text{PRED 'late'}]\} \\ \text{PRED} & \text{'arrive}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & [\text{PRED 'john'}] \end{bmatrix} \quad \begin{bmatrix} \text{REL} & \textit{late} \\ \text{ARG1} & \begin{bmatrix} \text{REL} & \textit{arrive} \\ \text{ARG1} & \textit{john} \end{bmatrix} \end{bmatrix}$$

Figure 2
Head-switching between f- and σ-structure.

this diagram, the semantic-structure corresponding to the entire f-structure has the adverb "late" as its top-level relation, even though this does not correspond to the syntactic (f-structure) head "arrive". Intuitively, the semantic argument (ARG1) of the adverb corresponds to the information coded in the partial f-structure (3), which comprises only the information concerning the subject and the predicate of the sentence.

(3) $\begin{bmatrix} \text{PRED} & \text{'arrive}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & [\text{PRED 'john'}] \end{bmatrix}$

The formal difficulty is that this is not an isolated unit of the f-structure and hence cannot be in the domain of σ. However, the f-structure description language can be extended by introducing a *restriction operator* which allows explicit reference to such smaller f-structures. The restriction operator "\" which is defined in (4) below[1] allows us then to refer to the partial structure (3) by the term $f\backslash\langle\text{ADJ } a\rangle$.

(4) The restriction operator is defined for an f-structure f and an attribute A by:
(i) $f\backslash\text{A} = f|Dom(f) - \{\text{A}\}$ if the value of $(f\ \text{A})$ is a structure, and
(ii) if $g \in (f\ \text{A})$ (i.e. if $(f\ \text{A})$ is set-valued) by

$$f\backslash\langle\text{A } g\rangle = \begin{cases} f\backslash\text{A} & \text{if } (f\ \text{A}) - \{g\} = \emptyset \\ f\backslash\text{A} \cup \{\langle\text{A}, (f\ \text{A}) - \{g\}\rangle\} & \text{else.} \end{cases}$$

[1] Cf. [Kaplan and Wedekind, 1993] for more details.

On the other hand, it becomes clear by examples with more than one adjunct that in addition a new source for recursion is needed, since it must in principle be possible to construct a multiple nesting for adjunct sets whose size is not bounded by any fixed finite upper bound.[2]

In order to identify this additional recursive device and to test our extended description language for adequacy, we picked out Montague semantics as a well-known semantic theory and tried to specify the syntax-semantics interface by a rule-based semantic interpretation mechanism. Our goal is to derive the semantic representation by analyzing the f-structure recursively. We assume an interpretation mechanism that operates on f-structures (possibly extended by information on the linear precedence of the predicates) and can be stated by very general compositionality principles without relying on any kind of transformations. This is because an f-structure of a sentence represents its deep structure in terms of predicate-argument relations where all information relevant for the interpretation is locally available. Furthermore, we want to ensure the "completeness" of the interpretation and to specify conditions which allow us to control the "conservativity" of the extension (for those who require that a theory of grammar such as LFG be strong enough to ensure the (semantic) well-formedness of the strings accepted by a particular grammar). On the other hand, we want the semantic structure to be accessible from the f-structure by an explicit interpretation function (σ-projector) in order to be able to formulate constraints, e.g. binding and scoping principles, which constrain the interpretation of the f-structures.

In this paper, we give three different type-driven interpretation mechanisms which fulfill the requirements given above. The first one is a rather simple top-down algorithm that can be described by our extended description language but cannot be used for all type systems. The second algorithm is a more powerful bottom-up algorithm which can be used for all type systems but not formulated in our description language. The third one, finally, is a top-down simulation of the second algorithm which is again describable in our description language. The fact that the third algorithm can be described by our extended description language seems to confirm the adequacy of our extension by the restriction operator. Furthermore, this investigation indicates that an additional *description-by-analysis* mechanism is needed within a codescription approach in order to handle cases

[2] This situation, where the recursion given by the context-free rule system turns out not to be the adequate or at least desirable carrier for specific (recursive) description purposes, is not unusual. Functional uncertainty was e.g. introduced as a new recursive device operating on f-structures, since unbounded dependencies could be more adequately handled by this new mechanism than by exploiting the recursive phrase structure rule system alone.

where the interpretation recursion is completely independent of the recursion given by the context-free grammar (cf. [Kaplan and Wedekind, 1993]).

2 A Simple Top-down Type-driven Interpretation Algorithm

In order to sketch how we can achieve a decomposition of an f-structure which is sufficient for its interpretation, we first introduce a simple top-down interpretation procedure which is restricted to certain special type systems. For the interpretation we generally assume from now on that *types* are assigned to all grammatical function values and semantically relevant atomic-valued features by a type assignment TY. Aside from the fact that grammatical functions and values and not c-structure constituents are typed, this assignment is similar to the one used in Montague grammar. The structure in figure 3 e.g. is an oversimplified typed f-structure[3] of the sentence

(5) John arrived late today.

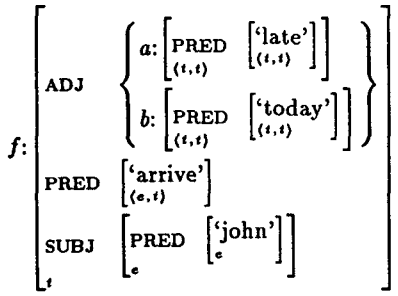

Figure 3
The typed f-structure of sentence (5).

The typing of the f-structures can e.g. be established by additional type-assigning annotations within a grammar. Examples of such augmented rules and lexical entries are given in (6).

(6) S → NP VP
$(\uparrow \text{SUBJ}) = \downarrow$ $\uparrow = \downarrow$
$TY(\downarrow) = e$ $TY(\uparrow) = t$

arrived: V, $(\uparrow \text{PRED}) = \text{'arrive'}$
...
$TY(\uparrow \text{PRED}) = \langle e, t \rangle$

It is of course possible to think of more sophisticated type inheritance mechanisms for the specification of the f-structure typing. The investigation of such mechanisms, however, is beyond our present concerns.

The restrictedness of the algorithm results from the fact that it operates under the assumption that we can recursively decompose each f-structure f into a substructure or set element which corresponds to a *one*-place main functor and the *rest* structure of f which is interpreted as the argument of that functor. Although this restriction seems to be rather strong, this algorithm gives the right hierarchical semantic structures for the problematic flat f-structures containing sentence modifiers. And if we assume the usual type-raising for the subcategorized functions, it also describes all possible structural ambiguities for predicate modifiers, quantifiers, etc.[4] In detail, the algorithm works as follows.

Proceeding from an initial mapping of a given f-structure f into an empty semantic structure the interpretation works top-down according to the following two principles:

If σg is defined and h is a substructure $(h = (g\ \text{A}))$ or an element of a set-value of g $(h \in (g\ \text{A}))$ and

(A1) $TY(g) = \tau$ and $TY(h) = \langle \tau', \tau \rangle$ then
(i) $(g\ \text{A}) = h \rightarrow (\sigma g\ \text{FU}) = \sigma h \wedge$
$(\sigma g\ \text{ARG}) = \sigma(g \backslash \text{A}) \wedge TY(g \backslash \text{A}) = \tau'$,
(ii) $h \in (g\ \text{A}) \rightarrow (\sigma g\ \text{FU}) = \sigma h \wedge$
$(\sigma g\ \text{ARG}) = \sigma(g \backslash (\text{A}\ h)) \wedge TY(g \backslash (\text{A}\ h)) = \tau'$,
(A2) $TY(g) = \tau \wedge TY(h) = \tau \rightarrow \sigma g = \sigma h$.

The principle (A1) allows us to select a substructure or an element of a set value of type $\langle \tau', \tau \rangle$ from a structure g of type τ, which is already mapped into the semantic representation, as a *functor* and interpret the *rest* of g as its argument which becomes then of type τ'.[5] If we apply principle (A1ii) to the structure in figure 3 and choose b as the functor we end up with the constellation in figure 4. For an interpreted structure g containing an immediate substructure or a set element h, principle (A2) drops the interpretation downwards if g and h are of the same type. This principle can then be applied e.g. to b of figure 4 and achieves the mapping in figure 5. Figure 6 gives a complete type-driven derivation of the functor-argument structure of (5) with wide scope of 'late'. One gets the other reading by first selecting b as described above.

Note that the *meanings* are not constructed by our procedure. The complete semantic representation results then from the insertion of the meanings of the basic expressions which are assumed to be specified in the lexicon via equations like the following:

late: ADV, $(\uparrow \text{PRED}) = \text{'late'}$
...
$\sigma(\uparrow \text{PRED}) = \lambda p(L(p))$.

[3] We drop the subcategorization frames in the following.

[4] For further illustration of the algorithms we give examples involving transitive verbs in the appendix.

[5] Note that a distinct re-interpretation of an already interpreted structure always fails, since predicates and predicate projections do not unify in LFG. Without this assumption, one would have to add to the principles the condition that g has no interpreted part.

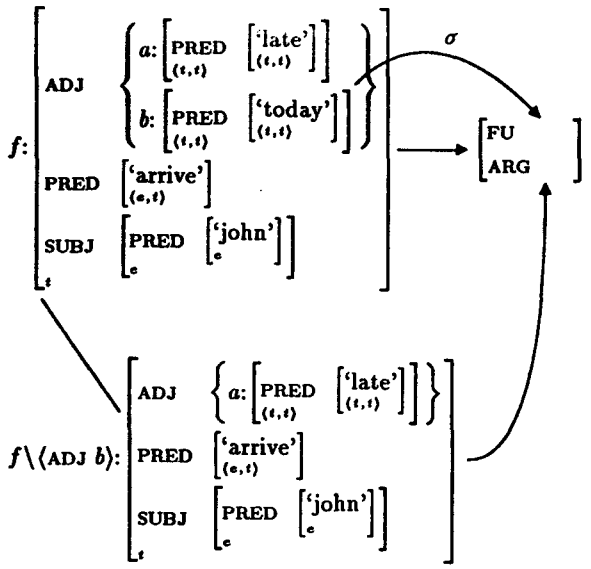

Figure 4
The result of applying principle (A1ii) to $b \in (f\ \text{ADJ})$ in figure 3.

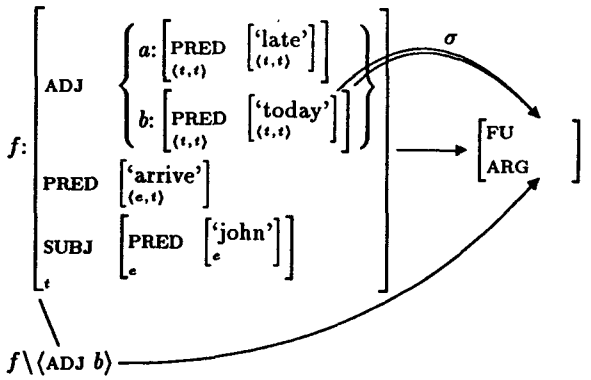

Figure 5
The result of applying principle (A2) to b in figure 4.

The result is then the following structure

$$\begin{bmatrix} \text{FU} & \lambda p(L(p)) \\ \text{ARG} & \begin{bmatrix} \text{FU} & \lambda q(T(q)) \\ \text{ARG} & \begin{bmatrix} \text{FU} & \lambda x(A(x)) \\ \text{ARG} & j \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

and the meaning of the sentence can be calculated bottom-up by λ-conversion in the usual way.

$$\begin{bmatrix} \text{FU} & \lambda p(L(p)) \\ \text{ARG} & \begin{bmatrix} \text{FU} & \lambda q(T(q)) \\ \text{ARG} & \begin{bmatrix} \text{FU} & \lambda x(A(x)) \\ \text{ARG} & j \\ A(j) & \end{bmatrix} \\ T(A(j)) & \end{bmatrix} \\ L(T(A(j))) & \end{bmatrix}$$

So, we end up with the expression $L(T(A(j)))$ which corresponds to the wide scope reading of 'late'.

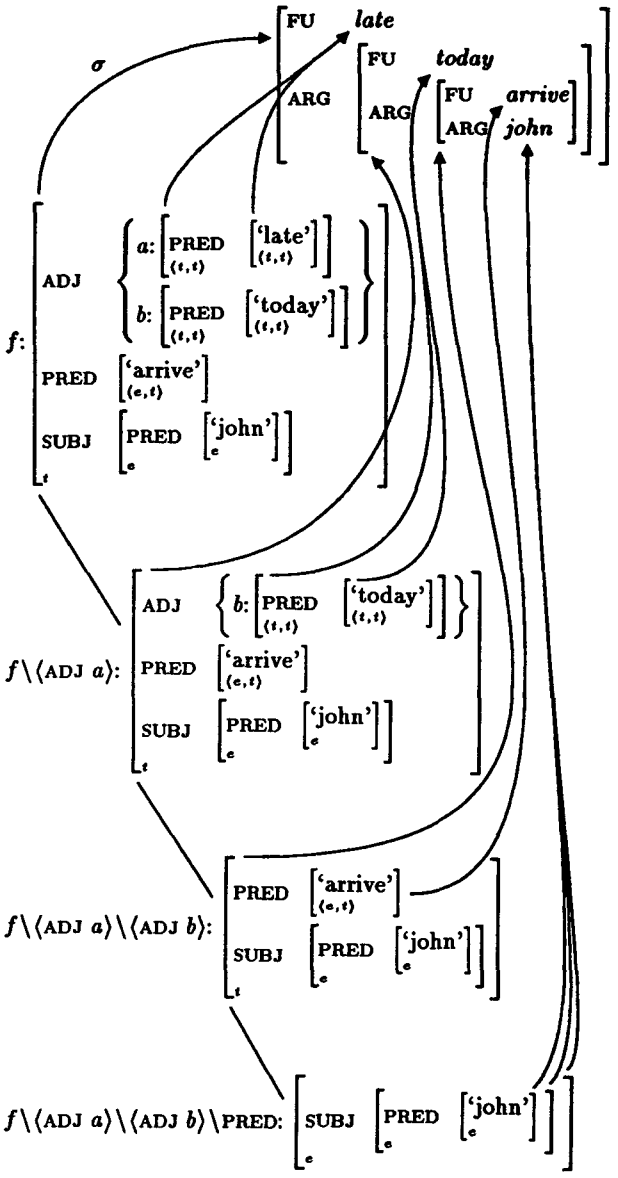

Figure 6
A complete derivation of one reading of (5).

3 A Bottom-up Type-driven Interpretation Algorithm

In the following we sketch a more powerful mechanism which can also handle cases where the functor is not given by a substructure (f A) or a set element $g \in (f\ \text{A})$ but by a partial structure g subsumed by f ($g \sqsubseteq f$) as e.g. in the typed structure for sentence (7) in figure 7. Here the part of the f-structure that comprises the modifiers and the predicate has to be interpreted as the main functor (either

3times(*twice*(*knock*)) or *twice*(*3times*(*knock*))).

(7) John knocked twice three times.

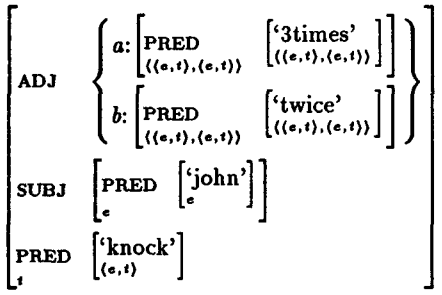

Figure 7
The typed f-structure of sentence (7).

Let "|" be a new operator which is defined for a substructure $(f\ \mathrm{A})$ of f by $f|\mathrm{A} = f\backslash\{\mathrm{A}\}$ and for a set element $g \in (f\ \mathrm{A})$ by $f|\langle \mathrm{A}\ g\rangle = \{\langle \mathrm{A}, \{g\}\rangle\}$. The value is simply the f-structure subsumed by f which has only attribute A with its value in f or with the singleton set-value g. For every attribute or attribute-element pair x, $f\backslash x$ and $f|x$ are in fact complementary with respect to f, that is, $f\backslash x \sqcap f|x = \emptyset$.

Proceeding from the interpretations of the basic expressions introduced by the lexical entries the algorithm works bottom-up according to the following principles:

(B1) If σh and σk are defined, $h \sqsubseteq g$, $k \sqsubseteq g$ and $h \sqcap k = \emptyset$ and $TY(h) = \langle \tau, \tau'\rangle$ and $TY(k) = \tau$, then
$\sigma h = (\sigma(h \sqcup k)\ \mathrm{FU}) \wedge \sigma k = (\sigma(h \sqcup k)\ \mathrm{ARG}) \wedge TY(h \sqcup k) = \tau'$.

(B2) If σh is defined and $TY(h) = \tau$, then

(i) $(g\ \mathrm{A}) = h \rightarrow TY(g|\mathrm{A}) = \tau \wedge \sigma h = \sigma(g|\mathrm{A})$, and

(ii) $h \in (g\ \mathrm{A}) \rightarrow TY(g|\langle \mathrm{A}\ h\rangle) = \tau \wedge \sigma h = \sigma(g|\langle \mathrm{A}\ h\rangle)$.

Principle (B2) pushes the interpretation from an interpreted structure h one level upwards to the partial structure of the given structure which contains only h as an attribute- or set-value and assigns to that partial structure the type of h. Note that principle (B2) can only be applied if $g|\mathrm{A}$ resp. $g|\langle \mathrm{A}\ h\rangle$ has no type assignment or is of the same type as h (otherwise the type assignment would not be a function).

If a structure g contains two disjoint partial structures h and k, one of them being an appropriate argument for the other, then the structures are interpreted according to principle (B1) as the functor resp. argument of the interpretation of their unification. This is then assigned the value-type of the functor. Figure 8 shows how the semantic representation of one reading of sentence (7) is constructed. We represent here attribute-value paths in DAG form

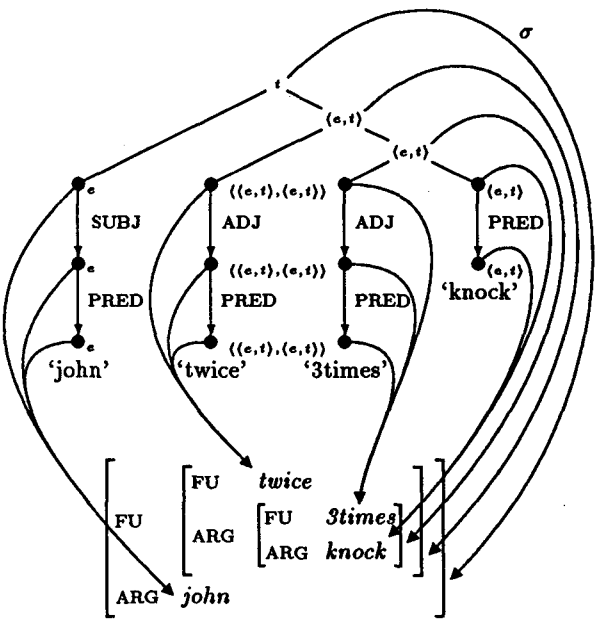

Figure 8
The bottom-up construction of one reading of (7).

and depict the decomposition of the f-structure as a graph where each subtree dominated by a branching node represents the partial f-structure which comprises the attribute-value paths contained in that subtree. The construction starts with the mapping of the terminal nodes provided by the lexical entries of the basic expressions. Each mapping of a structure dominated by a non-branching non-terminal node results from an application of principle (B2). The interpretation of a partial substructure (a structure dominated by a branching node) is constructed by principle (B1).

4 A Top-down Simulation of the Bottom-up Interpretation Algorithm

The restrictedness of the simple top-down algorithm results from the fact that the main functor was always assumed to take exactly *one* argument which is represented by the semantics of the rest of the f-structure. The algorithm fails in cases where the type of the substructure representing the main functor indicates that more than one argument is needed by the main functor in order to yield a meaning of the type of the entire f-structure. If we choose e.g. the '3times' modifier in the structure of figure 7 as the main functor (having widest scope), then we need a first argument of type $\langle e,t\rangle$ and a second argument of type e to get a meaning of type t. So, the rest of the structure corresponds in the general case to a list or set of arguments.

In order to overcome this difficulty, we assign to

the rest structure now a separate semantic structure. This structure is a set that contains typed variables for all those arguments which are still needed to saturate previously processed (complex) functors. If we start with the '3times' modifier this set contains the typed variables a_e and $a_{\langle e,t\rangle}$. In detail the algorithm works as follows.

If $TY(f) = \tau$ the algorithm starts from the initial assignment $\sigma f = f_\tau$ and proceeds top-down according to the following principles:

If σg is defined and h is a substructure ($h = (g\ \text{A})$) or an element of a set-value of g ($h \in (g\ \text{A})$), $TY(h) = \langle \tau_n, \langle \tau_{n-1} \ldots \langle \tau_1, \tau \rangle .. \rangle \rangle$ and

(C1) $\sigma g = k_\tau$ and $n > 0$, or there is a $k_\tau \in \sigma g$ and $n \geq 0$, then
$\sigma h = (k_\tau\ \text{FU}^n) = h_{\langle \tau_n, \langle \tau_{n-1} \ldots \langle \tau_1, \tau \rangle .. \rangle \rangle}$,
$(k_\tau\ \text{FU}^{i-1}\ \text{ARG}) = h^i_{\tau_i}$ (for each $i = 1, .., n$) and

(i) if $(g\ \text{A}) = h$ then
$$\sigma(g \backslash \text{A}) = \begin{cases} \{h^1_{\tau_1}, .., h^n_{\tau_n}\} \text{ if } k_\tau = \sigma g \\ (\sigma g - \{k_\tau\}) \cup \{h^1_{\tau_1}, .., h^n_{\tau_n}\} \text{ else,}^6 \end{cases}$$

(ii) if $h \in (g\ \text{A})$ then $\sigma(g \backslash \langle \text{A}\ h \rangle)$ is determined as in case (i),

(C2) $\sigma g = k_\tau$ and $n = 0$, then $\sigma g = \sigma h$.

In contrast to the simple top-down algorithm, each application of (C1) creates a new semantic structure which includes typed variables for all missing arguments. The new structure is linked to structures previously constructed either by explicit reentrancies or because they share common substructures. (The latter is enforced, since all those arguments (typed variables) which remain to be found after selecting k_τ are passed on to the semantic representation of the next restriction by (C1i,ii).) Reentrancies are used to link the (new) arguments to their right positions which are encoded in a functor-argument matrix in σg by applying (C1). Figure 9 gives three steps of a derivation of one reading of (7). (We omit in the example the upper indices of the typed variables provided by (C1), since no functor needs more than one argument of the same type.)

5 Completeness, Conservativity, Constraints and Compositionality

Since the meaning of a sentence with an f-structure f is given by the formula described by the semantic representation σf, the bottom-up construction is successful if we have constructed a value for σf. Within the top-down approaches the meaning of each basic expression represented in the f-structure has to be connected with the root σf, otherwise the semantic representation would not be a description of a well-formed formula.

[6] I.e., if $k_\tau \in \sigma g$.

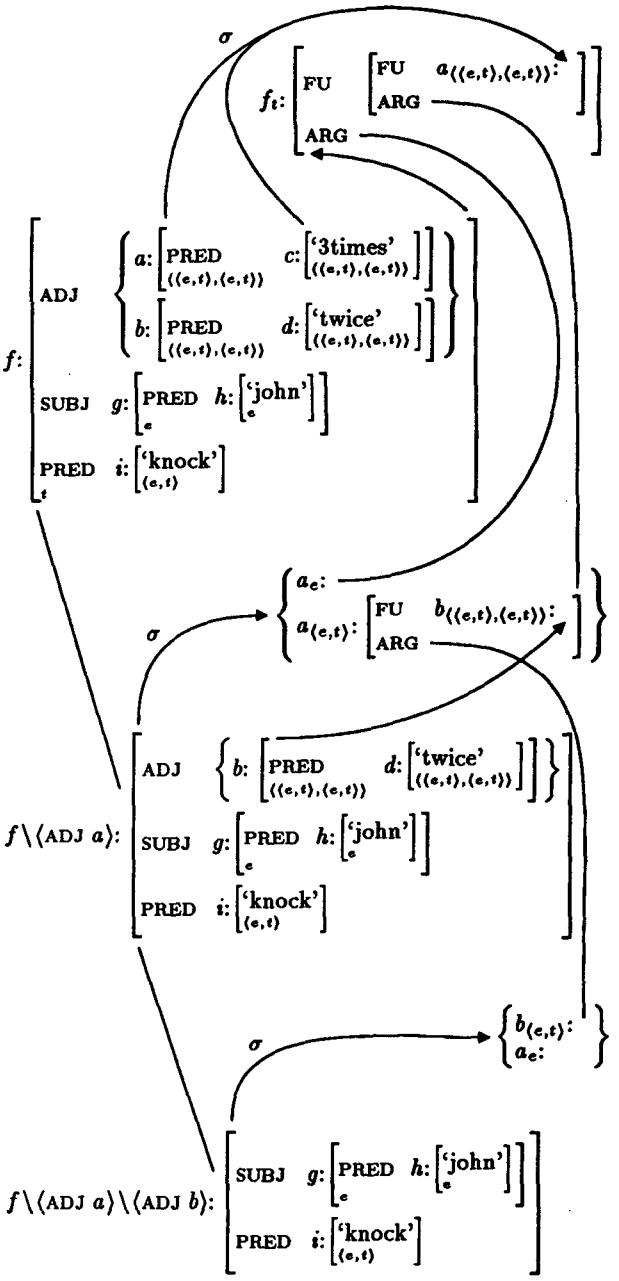

Figure 9
Some steps of the interpretation of (7).

In LFG, we can ensure that all syntactically well-formed utterances are also semantically well-formed by mechanisms which are already part of the theory. By the completeness and coherence conditions it can first be guaranteed that the different kinds of nuclei (consisting of a predicate and the functions it subcategorizes for) will get an interpretation of the right type. Since all free grammatical functions (ADJ) are homogeneous functors (argument and value are of the same type) and it is clear from the c-structure rules which type of argument they modify (a modifier on S-level is either a sentence or predicate modifier,

etc.), f-structures with free functions can also be ensured to be interpreted.

On the other hand particular readings can be excluded by global binding and/or scoping principles, similar to the ones formulated in [Dalrymple *et al.*, 1990]. These principles constrain the interpretation of the f-structures and their parts if special conditions are satisfied. By combining *outside-in* and *inside-out* functional uncertainty we can express by the following constraint e.g. that under some conditions Ξ the substructure $(\uparrow \text{A})$ of an f-structure has wide scope over $(\uparrow \text{B})$:

$$\Xi \rightarrow ((\text{FU } \sigma(\uparrow \text{A})) \text{ ARG}^{+} \text{ FU}) =_c \sigma(\uparrow \text{B}).$$

Due to the interpretation function (σ) between a typed f-structure and its semantic representation it is also possible to formulate a compositionality principle very similar to the classical one. The classical compositionality principle (Frege principle) says roughly that the meaning of an expression is a function of the meaning of its components and their mode of combination and is normally specified in terms of c-structure categories and their immediate constituents. As is well-known, the attractiveness of this principle gets lost to some degree if we have to handle phenomena which can only be described by assuming transformations on the constituent structures.

In LFG, the f-structures describe explicitly the underlying predicate-argument structures of well-formed expressions, and the components of an expression are taken to be the sets of string elements that are mapped via ϕ (the structural correspondence between c- and f-structure) to the units of its type-driven decomposed f-structure. On this view, the meaning of an expression remains a function of the meaning of its components. Thus, the reading of sentence (5) given in figure 6, e.g., is composed of the meanings of the components $\{\langle 1, \text{John}\rangle, \langle 2, \text{arrived}\rangle, \langle 4, \text{today}\rangle\}$ and $\{\langle 3, \text{late}\rangle\}$ associated with $f \setminus \langle \text{ADJ } a\rangle$ and a by ϕ, respectively. Their mode of combination (determined by the type assignment) is encoded in the functor-argument matrix as function-application of σa to $\sigma(f \setminus \langle \text{ADJ } a\rangle)$ (i.e. the meaning is $\sigma a(\sigma(f \setminus \langle \text{ADJ } a\rangle))$). Ambiguities result then from the indeterminism of the type-driven decomposition of the f-structure of a sentence. Thus, we can state for LFG a compositionality principle without assuming any kind of transformations, since all information relevant for the interpretation is locally available (cf. e.g. [Bresnan *et al.*, 1982]).

6 Conclusion

In this paper and in [Kaplan and Wedekind, 1993] we introduced a new formal device, the "restriction operator", into the language of functional descriptions. This operator provides a natural account of the misalignment between f-structures and semantic structures in cases where semantic units correspond intuitively to subsets of functional information. We tested this new descriptive device by formulating universal interpretation principles that derive representations of a Montagovian semantics by recursively analyzing typed f-structures. We outlined three interpretation algorithms, all of which depend on a *description-by-analysis* recursion that is independent of the recursion of the phrase-structure grammar. The first algorithm is formulated in terms of the f-designators provided by our extended description language, but is restricted to special type systems. In order to cover arbitrary type systems, we introduced a more powerful bottom-up algorithm which we were then able to simulate in a top-down fashion using again only the f-designators of our extended description language. This provides some support for the adequacy of our extended description language and reinforces the results reported by Kaplan and Wedekind [1993]. They combined the restriction operator with description-by-analysis rules for the interpretation of sentential and VP-adjuncts in a *codescription* approach which, although not explicitly driven by types, is patterned after the top-down algorithms presented here.

Appendix

Because of the particularly interesting cases of misalignment that they present, we concentrated in the main part of this paper almost exclusively on the interpretation of modifiers. Also, modifiers are involved in the head-switching translation problems discussed by Kaplan and Wedekind [1993], and the algorithms proposed here underlie the *description-by-analysis* approach that we developed in that paper. As suggested by the reviewers, we briefly sketch in this appendix how our approach extends to examples involving transitive verbs. Although their interpretation depends crucially on the type system and the actual meaning assigned to transitive verbs in the lexicon, we assume here a type system which is well-known from PTQ. This type system allows us to cover quantifier scope ambiguities as in (8)

(8) Every man loves a woman.

If we apply the simple top-down algorithm to the typed f-structure of (8) we get both readings due to an indeterminism in picking out the functors. Figure 10 shows some steps of a derivation where the universal quantifier has wide scope. (The other reading would be the result if $(f \text{ OBJ})$ were selected first.) Although the given type system would, of course, always yield SUBJ/OBJ scope ambiguities, specific readings can be excluded by a system of interacting scoping constraints, since the semantic structure is accessible via the σ-projector.[7] The functional uncer-

[7] In the few cases where the the scope is determined by the transitive verb itself, e.g. some passive forms in English, the appropriate reading can be enforced directly by using λ-expressions which refer explicitly to the mean-

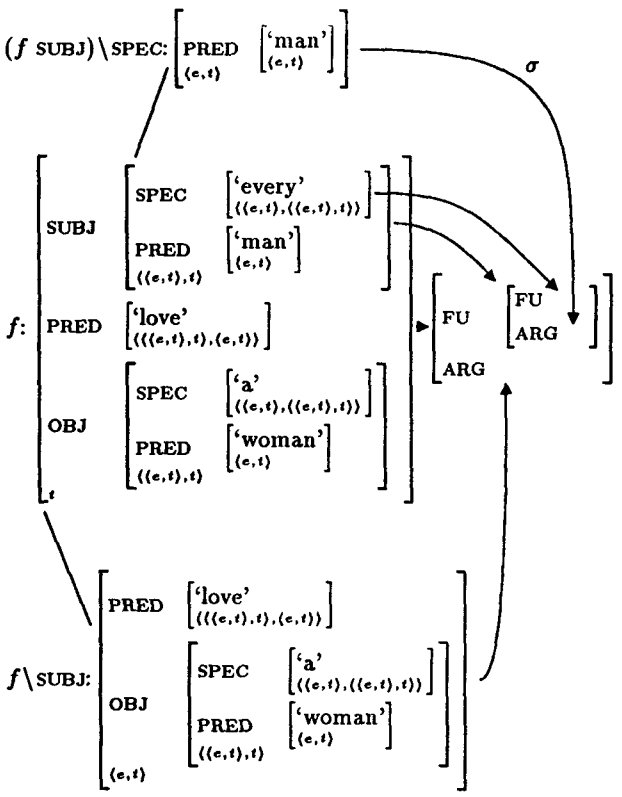

Figure 10
Some steps in the derivation of one reading of (8).

tainty constraint

$$((\text{FU}\ \sigma(\uparrow\ \text{SUBJ}))\ \text{ARG}^{+}\ \text{FU}) =_c \sigma(\uparrow\ \text{OBJ})$$

which can, appropriately annotated, be used in the grammar to enforce directly wide scope of the subject is just a simple example of the form of a constraint contained in such a system.

References

[Bresnan *et al.*, 1982] Bresnan, J., R. Kaplan, S. Peters, and A. Zaenen. Cross-serial Dependencies in Dutch. *Linguistic Inquiry* 13, 613–635, 1982.

[Dalrymple *et al.*, 1990] Dalrymple, M., J. Maxwell, and A. Zaenen. Modeling Syntactic Constraints on Anaphoric Binding. In *Proceedings of the 13th International Conference on Computational Linguistics.* Helsinki, 1990.

[Halvorsen, 1987] Halvorsen, P.-K. Situation Semantics and Semantic Interpretation in Constraint-based Grammars. CSLI Report No. 87-101. Stanford University, 1987.

ings of the grammatical functions. (If we would e.g. use $\lambda x(\text{OBJ}'(\text{love}'(x)))$ as the σ-value for 'love', we would always get wide scope of the subject.)

[Kaplan, 1987] Kaplan, R. Three Seductions of Computational Psycholinguistics. In P. Whitelock et al., eds., *Linguistic Theory and Computer Applications.* London: Academic Press, 1987.

[Kaplan and Bresnan, 1982] Kaplan, R., and J. Bresnan. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan, ed., *The Mental Representation of Grammatical Relations.* Cambridge, Mass.: The MIT Press, 1982.

[Kaplan *et al.*, 1989] Kaplan, R., K. Netter, J. Wedekind, and A. Zaenen. Translation by Structural Correspondences. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics.* Manchester, 1989.

[Kaplan and Wedekind, 1993] Kaplan, R., and J. Wedekind. Restriction and Correspondence-based Translation. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics.* Utrecht, 1993.

[Sadler and Thompson, 1991] Sadler, L., and H. Thompson. Structural Non-Correspondence in Translation. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics.* Berlin, 1991.

[Wedekind, 1988] Wedekind, J. Transfer by Projection. Ms., University of Stuttgart, 1988.