Generation as Structure Driven Derivation

Jürgen Wedekind
Institute for Natural Language Processing
University of Stuttgart
FRG

Abstract

This paper describes two algorithm which construct two different types of generators for lexical functional grammars (LFGs). The first type generates sentences from functional structures and the second from semantic structures. The latter works on the basis of extended LFGs, which contains a mapping from f-structures into semantic structures. Both algorithms can be used on all grammars within the respective class of LFG-grammars. Thus sentences can be generated from input structures by means of LFG-grammars and the same grammar formalism, although not necessarily the same grammar, can be used for both analysis and synthesis.

1 Introduction

Using the same grammar formalism, or even the same grammar, for both analysis and synthesis is usually regarded as an elegant, efficient and sometimes even as the psychologically most plausible approach to natural language parsing and generation. In this paper we want to show that this approach can be realized within the LFG-framework by defining two generation algorithms. The first permits the construction of generators for LFGs which generate sentences from functional structures. The second constructs generators which generate sentences from semantic structures.

Both algorithms are based on concepts of derivation for LFGs which can be strengthened in such a way that the derivation can be driven by a given input structure. The principles which underlie the control mechanisms for the derivation are sufficiently general to also be applicable to other unification based formalisms which allow the derivation of functional and/or semantic structures in parallel to constituent structures (e.g. PATR (cf. e.g. Shieber 83), Karttunen 86)).

For the generation from semantic structures, a derivation concept of this type can be defined, if, following a proposal by Halvorson and Kaplan (cf. Halvorson 87, Kaplan 87), projector equations are used to describe (co-)describe the semantic structure of a sentence. Since the projector mechanism is independent of the specific type of semantic theory, the algorithm works as long as this mechanism is used to build up semantic representations.

In addition, the derivation driven by the semantic structure can be directed by constraints over the f-structure. Usually, semantic information by itself is regarded as providing a basis for generation which is too weak to capture relevant distinctions in the surface form. This additional mechanism could contribute to make the generation more sensitive to syntactic and pragmatic information. Thus, additional functional information can enforce a specific syntactic realization, such as possessive marking, topicalization, and pragmatic realization.

It should be clear that the specific constraints which have to be or can be imposed on this kind of generation are subject to empirical studies on questions of syntax, discourse and dialogue. Similarly, for machine translation, one might consider experimental research on these subjects to establish what relevant functional information can be drawn directly from the f-structures of the source text, in order to guarantee a coherent target text. These important empirical questions will not be addressed directly in this paper. We consider it an essential requirement for an adequate generation algorithm, however, that it provides the respective possibilities of control.

2 Generation from functional structures

In this section we describe an algorithm which constructs a generator for an arbitrary given LFG which generates terminal strings from functional structures. Such an algorithm has to define for every LFG a relation \( \Gamma_s(\Phi, \Psi) \) (\( \Psi \) is generate from \( \Phi \)) between directed acyclic graphs (DAGs) and functional structures. The relation \( \Psi(\Phi, \Psi) \) is derivable from the start symbol \( S \) and \( \Psi \) has f-structure \( \Phi \).

\[
\forall \Phi, \Psi \in \text{DAG} \quad \text{Der} \quad \exists \Phi, \Psi \quad \text{if and only if} \quad \Phi \text{ is derivable from the start symbol } S \text{ and } \Phi \text{ has f-structure } \Psi.
\]

For the generation from semantic structures, a derivation concept of this type can be defined, if, following a proposal by Halvorson and Kaplan (cf. Halvorson 87, Kaplan 87), projector equations are used to describe (co-)describe the semantic structure of a sentence. Since the projector mechanism is independent of the specific type of semantic theory, the algorithm works as long as this mechanism is used to build up semantic representations.

In addition, the derivation driven by the semantic structure can be directed by constraints over the f-structure. Usually, semantic information by itself is regarded as providing a basis for generation which is too weak to capture relevant distinctions in the surface form. This additional mechanism could contribute to make the generation more sensitive to syntactic and pragmatic information. Thus, additional functional information can enforce a specific syntactic realization, such as possessive marking, topicalization, and pragmatic realization.

It should be clear that the specific constraints which have to be or can be imposed on this kind of generation are subject to empirical studies on questions of syntax, discourse and dialogue. Similarly, for machine translation, one might consider experimental research on these subjects to establish what relevant functional information can be drawn directly from the f-structures of the source text, in order to guarantee a coherent target text. These important empirical questions will not be addressed directly in this paper. We consider it an essential requirement for an adequate generation algorithm, however, that it provides the respective possibilities of control.

2 Generation from functional structures

In this section we describe an algorithm which constructs a generator for an arbitrary given LFG which generates terminal strings from functional structures. Such an algorithm has to define for every LFG a relation \( \Gamma_s(\Phi, \Psi) \) (\( \Psi \) is generate from \( \Phi \)) between directed acyclic graphs (DAGs) and functional structures. The relation \( \Psi(\Phi, \Psi) \) is derivable from the start symbol \( S \) and \( \Psi \) has f-structure \( \Phi \).

\[
\forall \Phi, \Psi \in \text{DAG} \quad \text{Der} \quad \exists \Phi, \Psi \quad \text{if and only if} \quad \Phi \text{ is derivable from the start symbol } S \text{ and } \Phi \text{ has f-structure } \Psi.
\]

Depending on what the adequacy condition \( \text{Ade} \) for this relation is, one will get different adequacy criteria for possible explications of what 'generation' can mean within the LFG-framework. We started from the - perhaps too idealized - condition that is normally used for the relation between strings and f-structures \( \Delta_s \), specified by an adequate parsing algorithm for LFGs:

\[
\forall \Phi, \Psi \in \text{DAG} \quad \text{Der} \quad \exists \Phi, \Psi \quad \text{if and only if} \quad \Phi \text{ is derivable from the start symbol } S \text{ and } \Phi \text{ has f-structure } \Psi.
\]

If we use this condition and insert it into the schema \( \text{Ade} \), then the relation \( \Delta_s \) is simply defined inversely by an adequate generation algorithm \( \Delta_s = \Delta_s(\Phi, \Psi, S) \) which satisfies this condition is simply a parser or transducer for the set of well-formed f-structures, which constructs for an input structure the set of all sentences which have this structure as their f-structure. This implies that the generator produces no output, if the input structure is not a well-formed f-structure.

2.1 Derivational background

The algorithm can be based on derivation concepts for LFGs which can be strengthened in such a way that the derivation can be driven by an input structure (structure driven derivation). However, the derivability conditions formulated in [Kaplan/Bresnan 82] cannot be used directly. According to [Kaplan/Bresnan 82] a terminal string \( s \) is regarded as well-formed if it satisfies the following conditions:

\[
\text{WFF}.
\]

1. There is a c-structure \( \phi \) for \( s \) that can be derived by the context-free base of the grammar.
2. There is an f-structure \( \phi \) and a mapping \( \phi \) from the c-structure nodes to the nodes of \( \phi \) such that \( \phi \) is the unique minimal f-structure that satisfies the annotations associated with the c-structure nodes. (The f-description solution algorithm (fda-algorithm) constructs both \( \phi \) and \( \phi \).)
3. All constraints in the f-description are satisfied by \( \phi \).
4. \( \phi \) is complete and coherent.

These conditions are tested in the following order: 1. -< 2. -< 3. -< 4. Thus, if the f-structure is built up after the derivation of the c-structure, it is impossible to use the functional information contained in an input structure for the control of the derivation of the c-structure. A decidable generation procedure presupposes the possibility of comparing the input structure with the partial f-structure of a derived partial c-structure. Thus, in order to drive the c-structure derivation by a given input structure it is necessary to derive the partial f-structure in parallel to a partial c-structure. This means that one can only use those derivation concepts which make f-description solutions for partial c-structures available in each step of the derivation. 

The concept on which the following algorithm is based is described in more detail in [Wedekind 86]. According to this concept, a derivation is a sequence of quadruples \( (c, \phi, \phi, c) \). Each quadruple consists of

\[
\begin{align*}
&\text{a partial c-structure}, \\
&\phi \text{ a partial f-structure}, \\
&\phi \text{ a mapping from the c-structure nodes into the set of nodes of } \phi, \text{ and} \\
&c \text{ a constraint set.}
\end{align*}
\]

(\( \phi \) and \( \phi \) would be the result of the fda-algorithm if it were applied to the corresponding annotated c-structure. \( c \) corresponds to the set of instantiated constraining equations contained in the f-description of \( c \). See the example in fig. 1.) I follow the usual convention of identifying the c-structure nodes with sequences of integers. The linear order of the edges of a tree is normally encoded by numbering the arcs, and every node is identified with the sequence of integers numbering the arcs along the path from the root to that node.
A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

\[ S \rightarrow \text{NP} \text{ VP VP} \rightarrow V \text{ VP} \rightarrow V \text{ SUBJ NUM} \]

The corresponding annotated f-structure is derivable by the following rules:

- \(c \rightarrow \text{NP} \rightarrow \text{VP} \rightarrow V \rightarrow \text{VP} \rightarrow \text{VP} \rightarrow \text{VP} \rightarrow \text{SUBJ NUM} \)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]

A grammar rule also introduces a quadruple.

\[ V \rightarrow (a, \Phi, \psi', C^\Psi) \]

The functional part \((\Phi, \psi')\) is obtained by applying the fds-algorithm to the annotated local tree represented by the rule, and by instantiating the metavariables in the constraining equations with the node indices of the local tree introduced by the rule. (The constraint set of a rule contains the metavariables in the constraining equations with the node indices of the root node \(s\), to which the S node is mapped by \(\phi_s\) and an empty constraint set \(C^\Psi\).)

The initial quadruple consists of an S-labeled root node \(c_s\), an f-structure root node \(\Phi_s\), to which the S node is mapped by \(\phi_s\), and an empty constraint set \(C^\Psi\).

\[ (c_s, \Phi_s, \psi', C^\Psi) \]
DEFINITION 2.2 A terminal string $s$ is generatable from an input structure $\Phi_{\text{in}}(\Gamma, \Sigma, \epsilon)$ iff there is a sequence $\omega_0 \ldots \omega_n$ such that:

- $\omega_0 = (\epsilon, \Phi_0, \phi_0, \Phi_{\text{in}}, h_0^0)$ and
- for all $i < n$, $\omega_i = (c', \Phi_i, \phi_i, \Phi_{\text{in}}, h_i^0)$ ($1 \leq i < n$) there is a generator rule $V \rightarrow \langle c, \Phi_i, \phi_i, C^j \rangle$ and
- $V$ is the label of a leaf $\mu$ of $c$.
- the substructure rooted by $h_i^0(\phi)$ is (aside from the $+\text{-labels}$) an extension of $\Phi_i$.
- for all $\langle p \rangle = (c, \Phi_i, \phi_i, \Phi_{\text{in}}, h_i^0)$ $\epsilon \Phi_i(D\Phi_i(p)) = \epsilon \Phi_i(h_i^0(\phi), p)$ etc.

- $c'$ is the result of expanding $\mu$ by $c$.
- $\Phi'$ is the minimal extension of $\Phi$ which results from $\Phi$ by unifying $\Phi_0$ with the substructure rooted by $\phi_0$.
- if $v$ is a node of $c$, whose $\phi$-value is $\delta'(q_{\text{in}}, p)$ in $\Phi$, then its $\phi'$-value in $\Phi'$ is $\delta'(q_{\text{in}}, p)$.
- if $\mu, j$ is a node of $c'$, not contained in $c$, the value of $\delta_j'$ is $\delta_j'(\phi_j, q_{\text{in}})$ and the value of $\phi$ for $\mu$ is $\delta'(q_{\text{in}}, p)$, then the value of $\delta_j'$ for $\mu, j$ in $\delta'(q_{\text{in}}, p)$ in $\Phi'$.
- for each node of the functional structure $\Phi_{\text{in}}$, of $\omega_i$ is $+\text{-labeled}$.
- $s$ is the terminal string of the $\epsilon$-structure of $\omega_n$.

Lemma 1 follows from the above.

**LEMA 1** $\forall \Phi : \text{DAG} \forall \nu \in \text{spect} \exists \nu^2(\Gamma, \epsilon, s) \Rightarrow \Delta_0(\epsilon, s) \Phi$

This lemma can easily be proved, since in each step of the generation of a sentence the applied rule can be applied exactly in the same way in the corresponding derivation step of a derivation of that sentence (and vice versa). So the substructure which includes all $+\text{-labeled}$ nodes of a generated functional structure corresponds exactly to that partial $\epsilon$-structure which is derived up to that step (and vice versa). Thus, the derived $\epsilon$-structure is identical to the generated $\epsilon$-structure, the derived $\epsilon$-structure is equal to the generated $\epsilon$-structure and thus identical (isomorphic) to the input structure. Since the constraints in the constraint set of a derivation must in fact be the (instantiated) constraints of all rules applied during the generation, the input structure satisfies all constraints iff the derived structure does.

### 3 Generation from semantic structures

In this section we use the ideas described in section 2 to develop an algorithm that constructs generators which generate terminal strings from semantic structures. These ideas are applicable if we can ensure, that:

- a) the semantic structures are representable as DAGs,
- b) the only operation which is used to construct the semantic structures is the unification operation, and
- c) the semantic structure of a sentence can be built up simultaneously with the derivation of the $\epsilon$- and $\epsilon$-structure of that sentence.

That a) and b) can be ensured for most of the current semantic theories, like Montague Semantics (MS), Discourse Representation Theory (DRT) and Situation Semantics (SS), is illustrated, for example, in [Reyle 88] (DRT), [Halvorsen 87] (SS) and in works concerning categorial grammars which are augmented by a unification component.\(^{18}\)

Condition c) is satisfiable if we follow a proposal by Halvorsen who describes a possible extension of LFGs such that the semantic representation of a sentence can be "simultaneously described (co-)described with the functional structure" ([Halvorsen 87], p.9). Halvorsen extends the formalism to include a new type of equation, which is used to build up a semantic representation and establishes an additional (partial) mapping from the $\epsilon$-structure nodes into the node set of the semantic structure.

Since the semantic structures are represented as DAGs, we can use for the generation from semantic structures, a condition like (*) as an adequacy condition, which refers to a semantic ($\epsilon$-) structure instead of an $\epsilon$-structure.

\[\text{(*)} \forall \mathcal{T} \in \text{spect} \forall \nu \in \text{spect}(\Gamma, \epsilon, s) \Rightarrow \Delta_0(\epsilon, s) \Phi\]

Since this condition implicitly assumes that $\mathcal{T}_{\Phi}$ is determined by an adequate parsing algorithm, this extension of the formalism is neutral with respect to the problem of the correctness of the extension. This neutrality is desirable, since the algorithm should be definable independent of a specific semantic theory. The question whether a semantic component is (or should be) a creative or a conservative extension of the syntactic theory (LFG), on the other hand, depends crucially on the specific semantic theory and on the format of the rules which prescribe how the compositionality principles of the chosen theory are to be translated into this new type of equation.

According to condition (**), the generator constructed by the algorithm for an (extended) LFG is a parser or transducer for well-formed semantic structures which, in principle, constructs for an input structure all sentences which have this structure as its semantic structure, analogously to section 2. A semantic structure alone, however, is usually regarded as too poor with respect to the syntactic information relevant for an adequate generation result within a natural language system. We therefore integrate the additional possibility of driving the derivation by syntactic (functional) information. This is possible because the $\epsilon$-structure of a sentence is built up in parallel to the $\epsilon$-structure driven derivation of that sentence. If we assume that sentences $\omega_i$ and $\epsilon$-structures $\Phi_i$ are related to an input structure $\Omega$ according to the following schema:
we can ensure by additional constraints on the f-structures that only those sentences are generable whose f-structures satisfy these constraints. Thus, we can drive the derivation in a way which is more sensitive to syntactic information. We can ensure, for example, that only a passive realization or a realization with a specific topic/focus structure is generable. Since not every input structure has a surface realization with an f-structure that fulfills these additional constraints, these constraints can be highly creative from a formal point of view. Therefore, a generator which drives the derivation by a semantic structure with additional constraints and also satisfies condition (**), must be capable to loosen the f-structure constraints if no output sentence can be derived whose f-structure satisfies the constraints, i.e., to drop constraints successively until an output sentence can be derived. The control of such a dropping procedure is again dependent on the application domain and can be determined only by empirical investigations.

3.1 The derivation of semantic structures

According to Halvorsen's proposal, the semantic structures are constructible by means of additional equations which are formulated with a projector \( \sigma \) and \( \phi \) that can be prefixed to any expression denoting a functional domain \( us \) and can be determined only by empirical investigations. The derivation of semantic structures with additional constraints and also satisfies condition (**), must be capable to loosen the f-structure constraints if no output sentence can be derived whose f-structure satisfies the constraints, i.e., to drop constraints successively until an output sentence can be derived. The control of such a dropping procedure is again dependent on the application domain and can be determined only by empirical investigations.

\[ (1 \text{ PRED}) = \text{kick’ Vo} \]
\[ ((c \text{ ARG}) = (c, 0” OBJ)) \]

In Part A the two equations at the bottom are called inter-modular equations.

We will first give the syntax of this new type of equation. Let us assume a distinction between functional attributes \( \{(L)\} \) and values \( \{A(L)\} \) and semantic attributes \( \{(A)\} \) and values \( \{A(A)\} \) with all four sets pairwise disjoint. Then the set of semantic \( (\sigma) \) designators contains the simple \( (\sigma) \) designators \( \{(\sigma)\} \) and the complex \( (\sigma) \) designators. A complex \( (\sigma) \) designator is a term \( \sigma(q, q) \) consisting of a complex \( (\sigma) \) designator \( (1 or 1) \) followed by a possibly empty sequence of functional attributes, and a possibly empty sequence \( q \) of semantic attributes. The \( (\sigma) \) equations, then, are those equations which have a complex \( (\sigma) \) designator in the first argument position and a complex or simple \( (\sigma) \) designator in the second.

\[ \begin{align*}
\text{If we assume that} & \\
\text{i. the fda-algorithm is used to solve the projector equations and to construct the semantic structure, and} & \\
\text{ii. constraints are expressible}, & 21
\end{align*} \]

then we can add the following two conditions to the WFF-condition:

\[ (Wt \Phi) \]
\[ (Ww \Phi) \] and

5. There is a \( (\sigma) \) structure and a partial mapping \( \sigma \) from the f-structure nodes to the nodes of \( \Sigma \), such that \( \Sigma \) is the only minimal \( (\sigma) \) structure that satisfies the projector equations associated with the c-structure nodes. (The fda-algorithm constructs also \( \sigma \) and \( \Sigma \).)

6. All \( (\sigma) \) constraints are satisfied by \( \Phi \), \( \Phi \), and \( \Sigma \). 22

These additional conditions must be tested in the order 3. \( \leq 5. \leq 6 \).

Since the description-solution algorithm is also used for solving the projector equations, it is easy to simulate the projector-mechanism by using an additional singular attribute. The solution algorithm enforces and preserves the function property (uniqueness) of the projector, i.e., if \( (\phi, p) \) and \( (\phi, p’) \) are terms designating some nodes in the f-structure which are mapped by \( \sigma \) to some nodes in the \( (\sigma) \) structure \( (\sigma(p, p’)) \), then the identity of \( (\phi, p) \) and \( (\phi, p’) \) which might be established by some other equations, will also enforce the identity of \( (\sigma(p, p)) \) and \( (\sigma(p, p’)) \). Thus, we can simulate the projector by using a singular attribute \( \sigma \) (not contained in \( L_0, A_0, L_1 \), and \( A_1 \)) that is inserted between the f-designator and the semantic attributes of a semantic designator according to the following rewrite schemata.

\[ (\sigma(1 p) g) \Rightarrow (1 p \sigma q) \]
\[ (\sigma(1 p) g) \Rightarrow (1 p \sigma q) \]

The identity of \( (\phi, p) \) and \( (\phi, p’) \) will then also enforce the identity of the values of \( (\phi, p \sigma) \) and \( (\phi, p’ \sigma) \).

If we simulate the projector mechanism in this way, the derivation of a \( (\sigma) \) structure in parallel to a c- and f-structure closely resembles the derivation process described in section 2.1. A rule introduces a quadruple, where \( \Psi \), is replaced by the DAG \( \Psi \), which is obtained by solving the functional and \( \sigma \)-annotations of the local tree introduced by the rule. The constraint set \( C^f \) contains the instantiated f- and \( \sigma \)-constraints expressed by that rule.

\[ V \rightarrow (c, \Psi, \Phi, C^f) \]

Fig. 7 A gives a slightly simplified example of a lexical entry from [Halvorsen 87], written as the expansion of a lexical category symbol.

\[ \begin{align*}
\text{In Part A the two equations at the bottom are called inter-modular equations.} & \\
\text{We will first give the syntax of this new type of equation. Let us assume a} & \\
\text{distinction between functional attributes \( (L) \) and values \( (A_L) \) and} & \\
\text{semantic attributes \( (A_S) \) and values \( (A_{AS}) \) with all four} & \\
\text{sets pairwise disjoint. Then the set of semantic \( (\sigma) \) designators} & \\
\text{contains the simple \( (\sigma) \) designators \( (\sigma) \)} & \\
\text{and the complex \( (\sigma) \) designators. A complex \( (\sigma) \) designator is} & \\
\text{a term \( \sigma(q, q) \) consisting of a complex \( (\sigma) \) designator (a} & \\
\text{metavariable \( (1 or 1) \) followed by a possibly empty sequence} & \\
\text{of functional attributes, and a possibly empty} & \\
\text{sequence \( q \) of semantic attributes. The \( (\sigma) \) equations, then, are} & \\
\text{those equations which have a complex \( (\sigma) \) designator in the first} & \\
\text{argument position and a complex or simple \( (\sigma) \) designator} & \\
\text{in the second.} & \\
\text{If we assume that} & \\
\text{i. the fda-algorithm is used to solve the projector equations and to} & \\
\text{construct the semantic structure, and} & \\
\text{ii. constraints are expressible,} & 21
\end{align*} \]

then we can add the following two conditions to the WFF-condition:

\[ (Wt \Phi) \]
\[ (Ww \Phi) \] and

5. There is a \( (\sigma) \) structure and a partial mapping \( \sigma \) from the f-structure nodes to the nodes of \( \Sigma \), such that \( \Sigma \) is the only minimal \( (\sigma) \) structure that satisfies the projector equations associated with the c-structure nodes. (The fda-algorithm constructs also \( \sigma \) and \( \Sigma \).)

6. All \( (\sigma) \) constraints are satisfied by \( \Phi \), \( \Phi \), and \( \Sigma \). 22

These additional conditions must be tested in the order 3. \( \leq 5. \leq 6 \). Since the description-solution algorithm is also used for solving the projector equations, it is easy to simulate the projector-mechanism by using an additional singular attribute. The solution algorithm enforces and preserves the function property (uniqueness) of the projector, i.e., if \( (\phi, p) \) and \( (\phi, p’) \) are terms designating some nodes in the f-structure which are mapped by \( \sigma \) to some nodes in the \( (\sigma) \) structure \( (\sigma(p, p’)) \), then the identity of \( (\phi, p) \) and \( (\phi, p’) \) which might be established by some other equations, will also enforce the identity of \( (\sigma(p, p)) \) and \( (\sigma(p, p’)) \). Thus, we can simulate the projector by using a singular attribute \( \sigma \) (not contained in \( L_0, A_0, L_1 \), and \( A_1 \)) that is inserted between the f-designator and the semantic attributes of a semantic designator according to the following rewrite schemata.

\[ (\sigma(1 p) g) \Rightarrow (1 p \sigma q) \]
\[ (\sigma(1 p) g) \Rightarrow (1 p \sigma q) \]

The identity of \( (\phi, p) \) and \( (\phi, p’) \) will then also enforce the identity of the values of \( (\phi, p \sigma) \) and \( (\phi, p’ \sigma) \).

If we simulate the projector mechanism in this way, the derivation of a \( (\sigma) \) structure in parallel to a c- and f-structure closely resembles the derivation process described in section 2.1. A rule introduces a quadruple, where \( \Psi \), is replaced by the DAG \( \Psi \), which is obtained by solving the functional and \( \sigma \)-annotations of the local tree introduced by the rule. The constraint set \( C^f \) contains the instantiated f- and \( \sigma \)-constraints expressed by that rule.

\[ V \rightarrow (c, \Psi, \Phi, C^f) \]

Fig. 7 A gives a slightly simplified example of a lexical entry from [Halvorsen 87], written as the expansion of a lexical category symbol.
root of that DAG. The derived semantic structure is the substructure of the derived DAG which is rooted by the value of the uppermost a-transition. This leads immediately to the definition of the derivability relation.

**Definition 3.1** A terminal string \( s \) is derivable with \( \sigma \)-structure \( (\Delta_u(s, \Sigma)) \) if there is a sequence \( w_0 \ldots w_n \) such that
- \( w_0 = (s, \Psi_1, \delta, \Sigma_0, h_1, C_2) \) and
- for all \( w_i = (c, \Psi_i, \delta, \Sigma_0, h_i, C_2) \), \( w_{i+1} = (c', \Psi_i', \delta', \Sigma_0', h_i', C_2') \) (\( 0 \leq i < n \)) there is a rule \( V \rightarrow (c, \Psi_i, \delta', \Sigma_0', h_i', C_2') \) and

- \( V \) is a label of a leaf \( \mu \) of \( e \)
- \( c' \) is the result of expanding \( \mu \) in \( e \) by \( c \)
- \( \Psi_i' \) is the extension of \( \Psi_i \) which results from \( \Psi_i \) by unifying \( \Psi_i \) with the substructure rooted by \( \delta_\mu \)
- if \( \nu \) is a node of \( e \), whose \( \delta' \)-value is \( \delta'(q_\nu, p) \) in \( \Psi_i \) then its \( \delta' \)-value in \( \Psi_i' \) is \( \delta'(q_{\nu'}, p) \)
- if \( \mu, j \) is a node of \( c' \), not contained in \( e \), the value of \( \delta_j \) is \( \delta_j^*(q_{\mu,j}, q) \) and the value of \( \phi \) for \( \mu \) is \( \delta'(q_{\nu}, p) \) then the value of \( \phi \) for \( \mu, j \) is \( \delta'^*(q_{\mu,j}, p, q) \) in \( \Psi_i' \)
- \( C_{\Psi_i'} \) contains besides \( C_{\Psi_i} \) the constraints \( (\mu, j, p) \equiv (\mu, i, q) \) etc. for all \( (j, p) \equiv (i, q) \) etc. in \( C_2 \)

- \( \Psi_i \) satisfies all constraints in \( C_2 \)
- \( \Psi_i \) is \( \{ \text{complete and } \} \)-coherent

\( \delta_{\Psi_i} \) satisfies the restrictions of \( C_{\Psi_i} \), \( \delta^*_{\Psi_i} \) is defined, \( \delta \equiv \delta_{\Psi_i} \)

- \( \Sigma_0 \) is the substructure of \( \Psi_i \) rooted by \( \delta_\mu(q_{\mu,j}, e) \)
- \( s \) is the terminal string of \( \sigma \)-structure of \( w_n \).

### 3.2 Generation as semantic structure driven and constrained derivation

The mechanisms for controlling completeness and coherence, which have been developed in section 2, together with the definition of the different types of derivation concepts can now be used to design an adequate algorithm for the generation from semantic structures, which in addition can be directed by functional constraints.

The process of generating from a semantic input structure \( \Sigma_m \) has to start with a complex entity

\[ (c_s, \Psi_1, \delta, \Sigma_m, h_s, C_2) \]

where \( \Psi_1 \) is an unordered tree consisting of only one a-transition whose value is \( \{ \text{a-labeled and dominates the unfolded input structure } \} \). \( \Sigma_m \). Fig. 9 shows the start entity for the generation from a simplified semantic input structure. \( h_s^* \) is the induced epimorphism from the substructure rooted by the \( \sigma \)-value of \( \Psi_1 \) onto \( \Sigma_m \). The start entity contains an empty set of constraints, since not all constraints expressed by the applied rules can be checked immediately (e.g. pure functional constraints) and have to be added to this set until they can be checked.

The derivation can be constrained by a set \( C_{\text{mod}} \) of pure functional and/or inter-modular constraints of the form

- \( (p, q) = (p, q) \)

which enforce the derivability of those sentences, whose \( f \) and \( \sigma \)-structures fulfill these constraints. (These constraints must eventually be retracted if no sentence is derivable.) E.g. the set

\[ C_{\text{mod}} = \{ (\Sigma \text{SUBJ } \sigma) = (\Sigma \text{ARG2}) \} \]

will enforce a passive realization of the input structure given in fig. 9 ("Peter is kicked by John"), while the set

\[ C_{\text{mod}} = \{ (\Sigma \text{BYOBJ } \sigma) = (\Sigma \text{ARG1}) \} \]

enforces a passive realization with the BYOBJ in the TOPIC-position ("By John, Peter is kicked").

The generator rules are constructed analogously to section 2 by an additional \( \{ \text{a-labeled of all nodes which are values of semantic attributes (not of the } \sigma \text{-attribute) } \} \) within the DAG-structure introduced by the rule. Fig. 10
two examples follows from the fact that these rules permit the derivation of structures with an unconnected semantic substructure, which becomes connected in later steps of the derivation by means of inter-modular equations. If a constellation is derived by means of these rules, as it is given schematically in fig. 11.

The generation by expansion of \( V \) proceeds without control by the input structure as long as the semantic structure introduced by \( V \) remains un-accessible. The coherence-condition is undermined and sentences can be generated whose semantic structure is not subsumed by the input structure.

The coherence-condition can be checked, however, if only those rules are applied which preserve the connectedness of the semantic structure derived so far. This restriction on the order-freeness of the rule application does not affect the adequacy condition (**), because in each derivation step of a sentence with a connected semantic structure there must be at least one node of the partial \( c \)-structure which is expandable in such a way that the expansion preserves the connectedness of the partial semantic structure.

Since it is possible that pure semantic constraints of the rule's constraint set refer to semantic substructures whose semantic function is uncertain up to that generation step, not all pure semantic constraints expressed by the rules can be tested immediately. This applies to pure semantic constraints which are constructable from inter-modular constraints by conversion. These constraints contain terms of the form \( (i \sigma p) \) (which correspond to \( (1 \sigma p) \) annotations) and they are not testable immediately if \( i \) is the value for \( \delta(p) \) (p non-empty) which corresponds to \( (1 \sigma p) \) annotation) and \( \delta_\sigma(p) \) has no \( \sigma \)-attribute. Hence, the pure semantic constraints cannot be tested in a particular step, the pure functional constraints, and the inter-modular constraints of the rule's constraint set have to be collected in \( C_\delta \). (The constraint-driven) generability relation has to then be defined as follows.

**Definition 3.2** A terminal string \( s \) is derivable from an input structure \( \Sigma_n \) under a constraint set \( C_{\text{mod}} \left( \Gamma_{\Sigma_m}(s, \Sigma, C_{\text{mod}}) \right) \) if there is a sequence \( w_0 \ldots w_n \) such that

- \( w_0 = (s, \Psi_1, \delta, \Sigma_n, h_1, C_2) \) and
- for all \( w_i = (c, \Psi_i, \delta, \Sigma_n, h_i, C_2) \), \( w_{i+1} = (c', \Psi_i', \delta', \Sigma_n, h_i', C_2') \) (\( 0 \leq i < n \)) there is a generator rule \( V \rightarrow (c, \Psi_i, \delta', \Sigma_n, h_i', C_2') \) and

- \( V \) is a label of a leaf \( \mu \) of \( e \)
- the substructure rooted by \( h^*(\delta(q_{\mu,j}, \sigma)) \) is (apart from the \( \sigma \)-labels) an extension of \( \Sigma_n \) (the substructure of \( \Psi_i \) rooted by \( \delta(q_{\mu,j}, \sigma) \))
- for all pure semantic constraints \( (j p) \equiv (i q) \) etc. in \( C_\delta \) for which \( \delta(q_{\mu,j}, \sigma) \) and \( \delta(q_{i,j}, \tau) \) is defined, \( \delta_{\Psi_i}(h^*(\delta(q_{\mu,j}, \sigma)), \tau) = \delta_{\Psi_i}(h^*(\delta(q_{i,j}, \tau)), \sigma) \) etc.
\( c' \) is the result of expanding \( c \) in \( c = \alpha \).

- \( \psi' \) is the minimal extension of \( \psi \) which results from \( \psi \) by unifying
  - \( \omega_\psi \) with the substructure rooted by \( \phi_\psi \)
  - \( \text{Vps} \psi'(v, \phi, \sigma, \eta) \text{Dom}(s) \rightarrow 3 \text{Vps} \psi'(v, \phi, s, \sigma) = 3 \psi'(v, \phi, s, \sigma) \)

  if \( \psi \) is a node of \( \omega_\psi \), whose \( \psi \)-value is \( \psi'(v, \phi, s, \sigma) \) in \( \psi_\psi \), then the \( \psi \)-value in \( \psi' \) is \( \psi'(v, \phi, s, \sigma) \).

  - if \( \phi \) is a node of \( \omega_\psi \), not contained in \( c \), the \( \phi \)-value of \( \psi' \) is \( \psi'(v, \phi, s, \sigma) \) and the \( \phi \)-value for \( \psi \) is \( \psi'(v, \phi, s, \sigma) \), then the \( \psi \)-value for \( \phi \) in \( \psi' \) is \( \psi'(v, \phi, s, \sigma) \).

- \( \text{C} + \) contains besides \( C \) the pure semantic constraints which cannot be tested in this step, the pure functional and the inter-modular constraints of \( C \) with \( \mu \) attached as a prefix to the node indices of the constraints

- \( \text{Vps} \psi'(v, \phi, \sigma, \eta) \text{Dom}(s) \rightarrow 3 \text{Vps} \psi'(v, \phi, \sigma, s) = 3 \psi'(v, \phi, \sigma, s) \)

  if each node of the substructure of \( \omega_\psi \) rooted by \( \omega(v, \phi, s, \sigma) \) (the semantic structure) is \( \psi \)-labeled

  - the substructure of \( \omega_\psi \) rooted by \( \omega(v, \phi, s, \sigma) \) is isomorphic to \( \Sigma_\omega \) (\( \Sigma_\omega \) is an isomorphism)
  - \( \psi_\omega \) is \( \psi \)-complete and \( \psi \)-coherent
  - \( \omega_\psi \) satisfies all constraints in \( C \)

  \( \psi \) satisfies all constraints in \( C \) and

  - \( s \) is the terminal string of the \( \psi \)-structure of \( \omega_\psi \).

If we define the (unconditional) generality relation \( \Gamma \) as

\[
\Gamma = \{ (\psi, \phi) \mid (\psi, \phi, s, \sigma, \eta) \}
\]

we can prove

**Lemma 2** If \( \omega_\psi \) DAG \( \mu \) V\( \omega_\psi \) \( \Gamma \) into \( \Delta_\omega (\Sigma) \)

The proof from left to right is carried out in analogy to Lemma 1. The right-to-left half follows from the fact that the order of the rule application during the derivation of a sentence with a connected semantic structure can always be rearranged in such a way that in each step of the derivation the partial semantic structure is connected. Otherwise the derived sentence would not have a connected semantic structure.

### 4 Acknowledgements

The research reported in this paper was partially supported by the EUROTRA-D accompanying research (BMFT grant No. 101 3207 G) and the ESPRIT project ACCORD (project P393). The author would like to thank Mark Johnson, Bob Kasper, Stefan Mamma, Klaus Netter, Klaus Reinhardt and the members of the working group on Unification at the University of Stuttgart (Inge Bethke, Jochen Dürr, Andreas Eisele and Henk Zeevat) who read carefully earlier versions of this paper and helped a lot in improving both the algorithm and its presentation. All remaining errors are of course my own.

### Footnotes

1. For parsing c.g. [Eisele/Dürr 86].
2. For the description of dialogue structures within LFG cf. [Netter 88].
3. The algorithms described here only work for those grammars which do not have regular expressions in the annotation (cf. [Kaplan/Maxwell/Zeevat 90]).
4. The algorithm has been implemented by Jochen Dürr and Stefan Mamma, cf. [Munma/Dürr 87].
5. This condition is weaker in different ways (cf. fn.1). However, since it depends on the domain of application how this is done in a particular case, and since we want to describe the algorithm independently of specific applications, we submit it to this strong condition.
6. Detailed information on solution algorithms can be found in [Johnsen 86] and [Johnson 87].
7. I was reminded by an anonymous reviewer to relate this paper to a paper by Black ([Black 86]), who must have had this procedure in mind, when he claimed that there is no way to derive structures from Extractions.
8. On the other hand, an LFG grammar, which does not allow transformations to be specified either as structural rules (e.g. [Uszkoreit 86]) and for unification grammars (UFG) cf. [Zeevat/Klein/Calder 87].
9. The extension of a syntactic theory by a semantic rule is called realised iff there are grammars where not all grammatically well-formed sentences are also semantically well-formed; otherwise the extension is called conservative.
10. Within a machine translation system this dropping of constraints could be controlled in such a way that a maximal similarity with respect to the stylistic features of the f-structure of the source language sentence has to be ensured. What 'maximal similarity' with respect to the stylistic features mean again presupposes some empirical work.
11. Since it is possible to express constraints referring to the mapping \( \sigma \) between \( \Phi \) and \( \Sigma \) (inter-modular constraints, for example), the reference to \( \Phi \) and \( \Sigma \) is necessary.
12. To separate the f- and a-structure from each other we only have to eliminate the \( \alpha \)-transitions.
13. The semantic function of a grammatical function is identified with the semantic attribute, whose value corresponds to the \( \psi \)-value of that grammatical function.

### References

[Black 86] Black, R.: Can a "Parsing Grammar" Be Used for Natural Language Gener-ation? The Negativa Example. In: Haskell, [ed.]: Proceedings of COLING 86, Bonn 1986.

[Block 86] Block, R.: Can a "Parsings Grammar" Be Used for Natural Language Gener-ation? The Negativa Example. In: Haskell, [ed.]: Proceedings of COLING 86, Bonn 1986.

[Bloch 87] Block, R.: Can a "Parsings Grammar" Be Used for Natural Language Gener-ation? The Negativa Example. In: Haskell, [ed.]: Proceedings of COLING 86, Bonn 1986.

[Bloch 87] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.

[Block 86] Block, R.: Can a "Parsings Grammar" Be Used for Natural Language Gener-ation? The Negativa Example. In: Haskell, [ed.]: Proceedings of COLING 86, Bonn 1986.

[Bloch 87] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.

[Block 86] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.

[Block 86] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.

[Bloch 87] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.

[Block 86] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.

[Block 86] Block, R.: A Logical Functional Grammar System in Prolog. In: Proceedings of COLING 87, Bonn 1987.