Some Remarks on the Decidability of the Generation Problem in LFG- and PATR-Style Unification Grammars

Jürgen Wedekind
Institute for Natural Language Processing
University of Stuttgart
Azenbergstr. 12
D-70174 Stuttgart, FRG
juergen@ims.uni-stuttgart.de

Abstract
In this paper, we prove the decidability of the generation problem for those unification grammars which are based on context-free phrase structure rule skeletons, like e.g. LFG and PATR-II. The result shows a perhaps unexpected asymmetry, since it is valid also for those unification grammars whose parsing problem is undecidable, e.g. grammars which do not satisfy the off-line parsability constraint. The general proof is achieved by showing that the space of the derivations which have to be considered in order to decide the problem for a given input is always restricted to derivations whose length is limited by some fixed upper bound which is determined relative to the "size" of the input.

1 Introduction
Unification Grammars with a context-free skeleton, like Lexical Functional Grammar (LFG) and PATR-II (cf. e.g. Kaplan and Bresnan 1982, Shieber et al. 1983) assign to a sentence not only a constituent structure (c-structure), but also an additional linguistic entity. In the rather restricted grammars of the early stage this entity is identified with a special graph structure, commonly called feature structure. Since a string is regarded as well-formed only if a (well-formed) feature structure is assigned to it by the grammar, two inverse decidability problems arise which had to be solved in order to know whether we can formulate terminating parsing and generation algorithms. If we retain the terminology of the early stages then an adequate parsing algorithm requires that we can decide for a given grammar and a given string whether there exists a feature structure assigned to it by the grammar (parsing problem) and an adequate generation algorithm requires that we can decide for a given grammar and a given feature structure whether there exists a sentence to which this structure is assigned by the grammar (generation problem).

While we already know for a long time that the parsing problem is undecidable (cf. Kaplan and Bresnan 1982, Johnson 1988), we want to show in this paper that the generation problem is decidable even for unrestricted (not off-line parsable) unification grammars. For the proof we first introduce in section 2 the type of grammar we want to consider. In section 3 we then define the generation problem and show its decidability in two steps.

2 Preliminaries
The unification grammars we want to consider consist of rules with a context-free skeleton and a set of annotations associated with the constituents mentioned in the rules. Typical examples taken from LFG and PATR-II are given in figure 1. For the formal definition of rules in LFG (left) and PATR-II format (right).

Figure 1
Examples of rules in LFG (left) and PATR-II format (right).
SG, 3RD) as individual constants. Furthermore, we assume for a context-free rule of the form \( A \rightarrow w \) (\( w \in (V_N \cup V_T)^* \)) that the variable \( x_0 \) is associated with \( A \) and that for each occurrence \( w_i \) in \( w \) there is a variable \( x_i \) which is associated with \( w_i \). For the formal reconstruction of LFG's we assume that each occurrence of \( \Downarrow \) in the annotation of \( w_i \) corresponds to an occurrence of \( x_i \) and that each occurrence of \( \Uparrow \) corresponds to an occurrence of \( x_0 \). For grammars in PATR-II format we suppose that occurrences of categories in the annotations correspond to the associated variables.

Before we give the definition of the grammars we want to investigate, we introduce the following notation. In the following we use \( S[x_1, \ldots, x_n] \) to indicate that the variables occurring in the set of formulas \( S \) are included in \( \{x_1, \ldots, x_n\} \) and \( S(x_1, \ldots, x_n) \) if the set of variables occurring in \( S \) is exactly \( \{x_1, \ldots, x_n\} \).

1. Definition. A unification grammar is a tuple \((V_N, V_T, S, F_1, V, \forall, R)\), consisting of a finite nonterminal vocabulary \( V_N \), a finite terminal vocabulary \( V_T \), a start symbol \( S \in V_N \) and a feature-description language \( L \) determined by a finite set of unary partial function symbols \( F_i \), a finite set of atomic values \( V \) and a denumerable set of variables \( \sigma \in \mathbb{N}^* \) with \( x_\sigma \neq x_{\sigma'} \) for \( \sigma \neq \sigma' \).

All vocabularies are pairwise disjoint. \( R \) is a finite set of rules of the form \( r = (A, w, S[x_0, \ldots, x_n]|(x_i \in V)\), \((A, w) \in V_N \times (V_N \cup V_T)^* \) (a context-free phrase structure rule) and \( S[x_0, \ldots, x_n]|(x_i \in V)\) a finite set of (quantifier-free) literals of \( L \).

According to our definition the LFG rules in figure 1 are now expressed as depicted in (1a) and the PATR-II rules as given in (1b). Note that the structure of the terms is now "mirror imaged", since we assume the attributes to be unary partial function symbols.

EXAMPLE 1. If we start a derivation \( \pi \) from \((S_0, \emptyset)\) and apply the S-rule in (1a) and the following VP-rule

\( \{VP \rightarrow V \text{ VP}', (x_0 \approx x_1, x_\text{COMP} x_0 \approx x_2)\} \)

we end up with the following sequence.

\( \pi_0 = (S_0, \emptyset) \) \( \pi_1 = (S_1, V_P_2, \emptyset), \{\text{SUBJ} x_0 \approx x_1, x_0 \approx x_2\} \) \( \pi_2 = (S_2, V_P_2, \emptyset), \{x_0 \approx x_1, x_\text{COMP} x_0 \approx x_2\} \) \( \vdots \)

For the steps depicted above the feature description is given by \( w_1 = \emptyset \) and \( w_2 = 2 \) and the feature description derived by \( \pi \) from 0 to 2 (\( S_0 \rightarrow S_2 \)) is

\( \{\text{SUBJ} x_0 \approx x_1, x_0 \approx x_2, x_2 \approx x_2, x_\text{COMP} x_2 \approx x_2\} \).

Sentences are then defined as follows.

2. Definition. A sequence of pairs \( \pi_0, \ldots, \pi_n \) with \( \pi_0 = (B, \emptyset) \) (\( B \in V_N \)) is called derivation of length \( n \) iff for each \( \pi_i = (B[i], \emptyset), S) \) \( (0 \leq i < n) \) there is a rule \( r = (A \rightarrow w_i \ldots w_k, S_c) \) such that

\[ \pi_{i+1} = (B[i], \emptyset), S_c) \]

In the definition we assume that the order of the arcs of a tree is encoded by numbering the arcs and that each node is identified with the sequence of integers numbering the arcs along the path from the root (\( \emptyset \)) to that node. In our bracket notation we add to a constituent its root node as the right and its root node label as the left index. In order to be able to refer to the c-structure derivation and to the sequence of feature descriptions and to have access to the nodes which are substituted in each step of a derivation, we define for a derivation \( \pi \) three other sequences.

3. Definition. Let \( \pi \) be a derivation of length \( n \). Then we define two sequences \( \omega \) and \( \gamma \) for each \( i = 0, \ldots, n \) with \( \pi_i = (T_c, S) \) by \( \omega_i = T_c \) and \( \gamma_i = S \) and a sequence \( \omega \) for each \( i = 1, \ldots, n \) with \( \omega_i = T_c \) and \( \gamma_i = S \).

Let \( S \) be a set of literals and \( \emptyset \) a unary partial mapping over the set of terms. Then the expression \( S[\emptyset] \) denotes the set of expressions obtained from \( S \) by simultaneously replacing each occurrence of a term \( \tau \) in each formula in \( S \) by \( \theta(\tau) \). The feature description derived by \( \pi \) is then defined by means of the following operation.

4. Definition. If \( \pi \) is a derivation of length \( n \) then the feature description derived by \( \pi \) from \( h \) to \( k \) \((0 \leq h < k \leq n)\) is given by

\[ S^\pi_{h \rightarrow k} = \bigcup_{i=h}^{k} \gamma_i \{\{x_j, x_m, j\} \mid x_j \text{ occurs in } \gamma_i\} \]

5. Definition. A terminal string \( w \) (\( w \in \mathbb{N}^* \)) is a sentence iff there is a derivation \((S_0, \emptyset) \) with \( \omega_n = s[w]\emptyset \) and \( \exists x_{\mu_1} \ldots x_{\mu_n} \land S^\pi_{n \rightarrow 0}(x_{\mu_1}, \ldots, x_{\mu_n}) \) satisfiable.

In the following we write \( S^\pi \) for \( S^\pi_{2n} \) if the interval covers the whole derivation, i.e. if \( \pi \) is of length \( n \).

Since a specific reduction algorithm and a few model-theoretic facts required in the proofs later on can be introduced by showing how satisfiability of such existential prenex formulas can be decided, we will continue with a short excursion on satisfiability.

---

1 The syntax and semantics of feature-description languages is given in the appendix.

2 A literal is an atomic formula or the negation of an atomic formula.

3 We use \( s[w]\emptyset \) to denote an S-rooted c-structure with yield \( w \).
2.1 Satisfiability

In order to test whether for a given finite set of literals \( S \) of a feature-description language \((2)\)

\[
(2) \forall x_1, \ldots, x_n \in S(x_1, \ldots, x_n)
\]

is satisfiable, we can exploit by skolemization well-known test procedures available for quantifier- and variable-free sets of such literals. Let \( C \) be a set of Skolem-constants \( \{x_1, \ldots, x_n\} \in \{C\} \) and \( \theta \) be a bijective function from \( \{x_1, \ldots, x_n\} \) to \( C \), then \((2)\) can be tested by testing the set of literals \((3)\) over \( L(C)\):

\[
(3) \exists[\theta],
\]

since \((2)\) and \((3)\) are equi-satisfiable. In the following we complete the procedure by introducing a reduction algorithm that reduces a set of literals \((3)\) according to a measure in a sequence of measure decreasing rewrite steps to a deductively equivalent set \((4)\) (in reduced form):

\[
(4) (S[\theta]),
\]

which is satisfiable if the terms \( \tau \) of all inequalities \( \tau \not\equiv \tau' \) of \((4)\) do not occur as subterms in equations of \((4)\).

For the proof we first introduce a few definitions and some notation. Let \( T \) be the set of a variable-free feature-description language \( L(C)\).

Then an injective function \( m : [T \rightarrow \mathbb{N}^*] \) is a measure iff it satisfies the following conditions for all \( \tau, \tau' \in T \) and \( \sigma \in F_1^*\):

(i) if \( |\tau| < |\tau'| \), then \( m(\tau) < m(\tau') \),
(ii) if \( m(\tau) \leq m(\tau') \), then \( m(\sigma \tau) \leq m(\sigma \tau') \).

For literals and sets of literals \( S \) we extend a measure \( m \) as usual by \( m((\tau \sim \tau') \tau) = m(\tau) + m(\tau') \) and \( m(S) = \sum \{m(\phi) \mid \phi \in S\} \).

In the following we use \( \tau \sim \tau' \) if \( m(\tau) \not> m(\tau') \) and \( \tau \not\approx \tau' \) to denote ambiguously \( \tau \not\approx \tau' \) or \( \tau \not\approx \tau \).

Let \( S \) be a set of literals then \( D \) denotes the set of all equations in \( S \) and \( \tau \in T \) is a term occurring in the formulas of \( S \) and \( \sigma \in F_1^* \).

SUB(T) is the set of all subterms of the terms in \( T \).

For the construction of a reduced form we need a specific partial choice function \( \rho \) which satisfies \( \rho(S) \in \{\tau \sim \tau' \tau \in T \mid \tau \in \text{SUB}([T \setminus \{\tau \sim \tau'\}])\} \) if the specified set is nonempty and undefined otherwise.

6. DEFINITION. For a given finite set of literals \( S \) and a choice function \( \rho \) we define a sequence of sets \( S \) by induction:

\[
S_0 = S
\]

\[
S_{i+1} = \left\{ \begin{array}{ll}
S_i \setminus \{\tau \sim \tau' \} & \text{if } \rho(S_i) = \tau \sim \tau' \text{ and } \rho(S_i) \in S \\
S_i & \text{if } \rho(S_i) \_ \text{ undefined}.
\end{array} \right.
\]

The feature-description language which in addition to \( L \) provides a distinct set of Skolem-constants \( C \). Cf. the appendix for more details.

The algorithm is adapted from Statman 1977 and Knuth and Bendix 1970 and first applied to feature-description languages by Beierle and Pletat (1988).

Since \( m(S_0) > m(S_{i+1}) \) if \( \rho \) is defined for \( S_i \), the construction terminates with a finite set of literals.

If we set

\[
S_p = S_{p_i}; \text{ with } t = \min\{i \mid S_{p_i} = S_{p_{i+1}}\}
\]

the following lemma can easily be proven by induction on the construction of \( S_p \).

7. LEMMA. For \( S_p \) it holds that:

(i) \( S \vdash S_p \),
(ii) if \( \tau \not\approx \tau' \in S_p \) then \( \tau \not\in \text{SUB}(T_{S_p \setminus \{\tau \not\sim \tau'\}}) \).

Since \( S_p \) is obviously not satisfiable if it contains an inequality \( \tau \not\approx \tau \) and \( \tau \) occurs as a subterm in \( E_p \), the whole proof is completed by showing that we can construct a canonical model satisfying \( S_p \) if \( S_p \) does not contain such an inequality.

For the model construction we need the set

\[
T_{E_p} = \{\tau \in \text{SUB}(E_p) \mid \exists \tau' (\tau \not\approx \tau' \in E_p)\}
\]

and the function \( h^c \in [\text{SUB}(T_{E_p}) \rightarrow T_{E_p}] \) which is defined for each \( \tau \in \text{SUB}(T_{E_p}) \) by

\[
h^c(\tau) = \begin{cases} 
\tau \tau' : (\tau \not\approx \tau' \in E_p) & \text{if } \tau \not\approx \tau' \in E_p \\
\tau & \text{otherwise}.
\end{cases}
\]

That \( h^c \) is well-defined results of course from \((7)\).

8. DEFINITION. For a set of literals \( S_p \) the canonical term model is given by the pair \( M_p = (U_p, \Theta_p) \), consisting of the universe

\[
U_p = \begin{cases} 
T_{E_p} & \text{if } E_p \not= \emptyset \\
\emptyset & \text{otherwise}
\end{cases}
\]

and the interpretation function \( \Theta_p \), which is defined for \( c \in V \cup C \), \( f \in F_1^* \) and \( \tau \in U_p \) by:

\[
\Theta_p(c) = \begin{cases} 
h^c(c) & \text{if } c \in \text{SUB}(T_{E_p}) \\
\text{undefined otherwise} & \text{otherwise}
\end{cases}
\]

\[
\Theta_p(f)(\tau) = \begin{cases} 
h^c(f \tau) & \text{if } f \tau \in \text{SUB}(T_{E_p}) \\
\text{undefined otherwise} & \text{otherwise}
\end{cases}
\]

For \( M_p \) which is well-defined the following lemma holds:

9. LEMMA. If \( \tau \) is a subterm of \( T_{S_p} \) then

(i) \( \Theta_p(\tau) = h^c(\tau) \), if \( \tau \in \text{SUB}(T_{E_p}) \),
(ii) \( \tau \in \text{SUB}(E_p) \), if \( \tau \in \text{Dom}(\Theta_p) \).

PROOF. (By induction on the length of \( \tau \).) The lemma is trivial for constants. By showing \((7)\) before \((6)\) we get the induction step for a subterm \( f \tau \) of \( T_{S_p} \) in both cases according to

\[
\Theta_p(f \tau) = \Theta_p(f)(\Theta_p(\tau)) = \Theta_p(f)(h^c(\tau)) = \Theta_p(f)(\tau).
\]

We get \( \Theta_p(\tau) = h^c(\tau) \) by inductive hypothesis and \( h^c(\tau) = \tau \), since \( \tau \not\in U_p \) would imply the existence of

\[47\]
\( \tau \Rightarrow \tau' \in E_p \) and \( f \tau \) could not be a subterm of \( T_p \), according to lemma 7(ii). Now, if (i) \( f \tau \in \text{SUB}(T_{E_p}) \) then \( \exists_p(f)(\tau) \) is defined and equal to \( h^c(f \tau) \) and (ii) if \( f \tau \in \text{SUB}(T_p) \) and \( \exists_p(f \tau) \) is defined then \( f \tau \in \text{SUB}(T_{E_p}) \). \( \square \)

On the basis of lemma 9 it is now easy to prove:

10. LEMMA. \( \forall \tau \neq \tau' \in S_p (\tau \not\subseteq \text{SUB}(T_{E_p})) \rightarrow \exists_{M_p} S_p \).

PROOF. (If the condition is satisfied \( \exists_{M_p} \phi \) holds for every \( \phi \in S_p \). If \( \phi = \tau \Rightarrow \tau' \in S_p \) with \( m(\tau') \leq m(\tau) \), then \( \tau' \in T_{E_p} \) by 7(ii) and hence \( h^c(\tau') = \tau' \). We get then \( h^c(\tau) = \tau' \) for \( m(\tau') = m(\tau) \) by \( \tau = \tau' \) and for \( m(\tau') < m(\tau) \) by the definition of \( h^c \), since \( \tau \not\subseteq T_{E_p} \). Thus \( \exists_p(\tau) = \exists_p(\tau') \) by 9(i) and hence \( \exists_{M_p} \phi \). Assume \( \phi = \tau \not\Rightarrow \tau' \). If \( \tau \neq \tau' \) were satisfied by \( M_p \), we would get \( \exists_p(\tau) = \exists_p(\tau') \) and by 9(ii) \( \tau, \tau' \in \text{SUB}(T_{E_p}) \). Since 7(ii) ensures \( h^c(\tau) = h^c(\tau') = \tau = \tau' \), we would have \( \tau \not\Rightarrow \tau \in S_p \) with \( \tau \in \text{SUB}(T_{E_p}) \). \( \square \)

Finally it should be mentioned that \( M_p \) is a unique (up to isomorphism) minimal model for \( S_p \), i.e. if \( M \) is a model for \( S_p \), homomorphic to \( M_p \), then every minimal submodel of \( M \) that satisfies \( S_p \) is isomorphic to \( M_p \).

3 The Generation Problem and its Decidability

Although it was not necessary for the definition of the sentences derivable by a unification grammar, we now have to make explicit that also a feature description is assigned to a sentence.

11. DEFINITION. A terminal string \( w \) (\( w \in V^*_T \)) is derivable with feature description \( \exists_x \ldots x \phi(x_1, \ldots, x_1) \) iff the feature description is satisfiable and there is a derivation \( (T_x, \emptyset) = \tau_0 \ldots \tau_n \) with \( \omega_\emptyset = s[w]_\emptyset \) and \( \phi = \Lambda \wedge S^* \).

Since deductively equivalent consistent feature descriptions are assumed to describe the same set of feature structures (models), the assignment of entities to terminal strings determined by a unification grammar is then formally given by a binary relation \( \Delta \) between terminal strings and sets of classes of deductively equivalent feature descriptions \( \{\exists_x \ldots x \phi(x_1, \ldots, x_1)\} \). \( \uparrow \)

12. DEFINITION. For each terminal string \( w \) and each class \( \{\exists_x \ldots x \phi(x_1, \ldots, x_1)\} \):

\[ \Delta(w, \{\exists_x \ldots x \phi(x_1, \ldots, x_1)\}) \text{ if } w \text{ is derivable with } \exists_x \ldots x \phi(x_1, \ldots, x_1). \]

Definition 12 now brings us closer to the problem, since we can for any unification grammar in rather abstract terms specify what parsers and generators are: a parser is a procedure which recursively enumerates for any given string \( w \) the set

\[ \{ \exists_x \ldots x \phi(x_1, \ldots, x_1) \mid \Delta(w, \{\exists_x \ldots x \phi(x_1, \ldots, x_1)\}) \} \]

3.1 Redundant Recursions and Pumping

For the proof that for a derivation of a sentence \( w \) with (consistent) feature description \( \phi \) there always exists a short derivation of a sentence \( w' \) with feature description \( \phi' \) and \( \phi \prec \prec \phi' \) whose length is bounded by the “size” of \( \phi \). By showing in the second step that two deductively equivalent consistent feature descriptions have the same “size” theorem 13 follows, since only a finite set of derivations (those whose length does not exceed this upper bound) have to be inspected in order to decide \( \exists w \in V^*_T (\Delta(w, [\exists_x \ldots x \phi(x_1, \ldots, x_1)])) \) for an arbitrary consistent input \( \phi \).

13. THEOREM. It is decidable for each feature description \( \exists y_1 \ldots y_k \psi(y_1, \ldots, y_k) \) whether there is a terminal string \( w \) which is derivable with \( \exists x_1 \ldots x_1 \phi(x_1, \ldots, x_1) \) and

\[ \exists y_1 \ldots y_k \psi(y_1, \ldots, y_k) \prec \prec \exists x_1 \ldots x_1 \phi(x_1, \ldots, x_1). \]

In the first step we show that we can always shorten a derivation of a sentence \( w \) with (consistent) feature description \( \phi \) to a derivation of a sentence \( w' \) with feature description \( \phi' \) and \( \phi \prec \prec \phi' \) whose length is bounded by the “size” of \( \phi \). By showing in the second step that two deductively equivalent consistent feature descriptions have the same “size” theorem 13 follows, since only a finite set of derivations (those whose length does not exceed this upper bound) have to be inspected in order to decide \( \exists w \in V^*_T (\Delta(w, [\exists_x \ldots x \phi(x_1, \ldots, x_1)])) \) for an arbitrary consistent input \( \phi \).

We assume here strong reversibility, since a generator for a given input \( y \) simply a parser which operates on \( \Delta^{-1} \): it recursively enumerates instead of \( \{ x \mid \Delta(x, y) \} \) the set \( \{ x \mid \Delta^{-1}(x, y) \} \).
In the first step, we eliminate as many variables as possible by substitution. The conversion is performed in two steps. In the second step, we make the set of literals independent, i.e., we remove those literals which are dependent, i.e., we remove those literals which are eliminable in \( S(x_1, \ldots, x_i) \).

If we assume that a given derivation of a sentence is already shortened to a derivation without redundant recursions it remains to show that the length of such a derivation could not exceed the upper bound determined by the “size” of the derived feature description.

The “size” of a consistent feature description is on the one hand determined by the size of its minimal normal form into which every feature description can be converted. The conversion is performed in two steps. In the first step, we eliminate as many variables as possible by substitution.

15. Definition. If \( S(x_1, \ldots, x_i) \) is a set of literals, then \( x_i \) is eliminable in \( S(x_1, \ldots, x_i) \) iff there is a term \( \tau \) not containing \( x_i \) such that \( \vdash \exists x_1, x_2, \ldots, x_i (\neg S(x_1, \ldots, x_i) \lor \tau) \).

16. Notation. In the following we write \( S(x_1, \ldots, x_i) \) iff each \( x_i \) is not eliminable in \( S \).

17. Definition. We assign to a set of literals \( S'(x_1, \ldots, x_i, x'_1, \ldots, x'_k) \) a set \( R(S') \) which contains a set \( S[x_1, \ldots, x_i] \) iff there is a substitution \( \theta \in \{ x'_1, \ldots, x'_k \} \rightarrow T(x_1, \ldots, x_i) \) such that

\[
\vdash \exists x_1, x_i, x'_1, \ldots, x'_k (\neg S' \lor \theta(x'_i))
\]

for all \( x'_i (1 \leq i \leq k) \) and \( S[x_1, \ldots, x_i] = S'[\theta] \).

By the substitutivity theorem we get:

18. Lemma. If \( S(x_1, \ldots, x_i) \in R(S'(x_1, \ldots, x_i, x'_1, \ldots, x'_k)) \) then \( \exists x_1, x'_k (S \equiv \theta(S')) \).

In the second step, we make the set of literals independent, i.e. we remove those literals which are implied by the remaining subset.

\[ T(x_1, \ldots, x_i) \] denotes the set of terms over \( \{ x_1, \ldots, x_i \} \) and \( F_1 \).

19. Definition. A set of literals \( S(x_1, \ldots, x_i) \) is independent iff there is no formula \( \phi \in S \) for which \( \vdash \exists x_1, x_i (\neg (S \setminus \{ \phi \}) \lor \phi) \) holds.

Normal forms are then defined as follows.

20. Definition. A consistent feature description \( \exists x_1, x_i S(x_1, \ldots, x_i) \) is in normal form (in the following indicated by a \( \nu \) index) iff \( S(x_1, \ldots, x_i) \) is independent.

Furthermore, we call \( \exists x_1, x_i S_\nu(x_1, \ldots, x_i) \) a normal form of \( \exists x_1, x_i, x'_1, \ldots, x'_k S'(x_1, \ldots, x_i, x'_1, \ldots, x'_k) \) iff \( S_\nu(x_1, \ldots, x_i) \) is an independent subset of \( S[\theta] \in R(S') \) and \( \exists x_1, x_i (\neg S \lor S'(x_1, \ldots, x_i)) \equiv \neg S'[\theta] \).

Lemma 18 and the condition in definition 20 ensure that a consistent feature description and its normal forms are deductively equivalent.

In order to be able to show the existence of a redundant recursion, we exploit the simple fact that the information which contributes a literal in a normal form with a minimal model \((U_0, \Phi_0)\) can be specified by an equation \( \tau \equiv \tau' \) where \( |\tau| + |\tau'| \leq |U_0| + 2 \). A literal \( \tau \approx \tau' \) \((|\sigma| > 0)\) whose terms are longer must always be reducible by a loop \( \sigma\tau' \approx \tau' \) to a shorter equation. Since the construction of such an information piece can be done with a substitution of some \( \theta \) fixed length, there must be a redundant recursion if the length of whole derivation exceeds a fixed value which is dependent on \(|U_0|\) and \(|S_\nu|\) and exactly specified in lemma 21.10

21. Lemma. Suppose that \( w \in V_\nu \) is derivable with \( \phi = \exists x_1, x_i x'_1, \ldots, x'_k S'(x_1, \ldots, x_i, x'_1, \ldots, x'_k) \) over \( \pi \) of length \( n \), then \( \exists x_1, x_i S_\nu(x_1, \ldots, x_i) \) is a normal form of \( \phi \) and \( M = (U_0, \Phi_0) \) is a minimal model of \( \phi \). If \( \pi \) has no redundant recursions then each path of \( \omega_n \) is shorter or equal to \(|V_\nu| \cdot (3|U_0| + 1) - |S_\nu| + 1 \).

Proof. Suppose, one path of \( \omega_n \) was longer than \(|V_\nu| \cdot (3|U_0| + 1) - |S_\nu| + 1 \), then more than \((3|U_0| + 1) - |S_\nu| + 1 \) different nodes on that path had to be labelled by the same \( \Delta \in V_\nu \). Without loss of generality we can assume that \( \pi \) is a derivation whose c-structure derivation \( \omega \) has the form \( \omega_0, \ldots, \omega_m, \omega_n \) with \( \omega_m = s[\mu, \Phi_0] \) for each node \( \mu \) on that path which is labelled by \( \Lambda \) (reorder if necessary). In order to exclude that complex inferences are used to build up \( \theta \), we assume further that \( \theta \) is non-deterministically constructed from \( S^\nu \) by recursive variable substitution, i.e. we require for each \((x, \sigma, \tau) \in \theta\) either \( x \approx \sigma \tau \in S^\nu \) or \( 3x \approx \sigma y \in S^\nu((y, \tau) \in \theta) \). Finally, let \( S \subseteq S^\nu \) with \( \Phi_0 = S[\theta] \). In order to identify the redundant recursion we have to consider the following cases.

1. Suppose there are more than \(|S_\nu| + 1 \) \( \Lambda \)-labelled nodes \( \mu \) such that \( x_\mu \) does not occur in \( S^\nu \), then there must be more than \(|S_\nu| \) non-overlapping recursions. For at least one of those recursions \( \pi_{m+1, \ldots, m+k+1} \) it must hold that

\[
(S \cap S^\nu_{m+1, \ldots, m+k+1}) \subseteq (S_\nu^m S_{m+k+1, \ldots, n}^\nu).
\]

10The given factor \(|V_\nu| \cdot (3|U_0| + 1) - |S_\nu| + 1 \) decreases for more restricted grammars, like e.g. grammars which allow only feature descriptions with single-rooted and/or acyclic minimal models.
But then
\[ S \subseteq (S_m^t \cup S_{m+k}^t) \]
and \( \pi_{m+1} \ldots \pi_{m+k} \) must be redundant.

2. If case 1 does not apply there must be more than 3 \(|S_w| - 1\) distinct A-labelled nodes \( \mu \) on that path such that \( x_\mu \) occurs in \( S^x \) and for more than 3 \(|S_w| + 1\) of these nodes must pairwise hold
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]
But then there must be at least three recursions such that
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]

We can then assign to each recursion \( \pi_{m+1} \ldots \pi_{m+k} \) (\( m = i, k = l; m = i + l, k = v \) or \( m = i + l + v, k = z \)) a type which corresponds to the strongest of the following conditions the recursion satisfies.

(a) \( \pi_{m+1} \ldots \pi_{m+k} \) satisfies
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]

(b) It holds only
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]
with \( |\sigma| > 0 \) and \( \sigma \theta(x_\mu) \approx \theta(x_\mu) \) is implied by
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset S_0^x \cup S_{m+1}^x \cup S_g) \]
where \( S_g \) is the set of ground literals of \( S \).

(c) Or it holds
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]
with \(|\sigma'| > 0\) and \( \sigma' \theta(x_\mu) \approx \theta(x_\mu) \) is implied by
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset S_0^x \cup S_g). \]

(d) If a recursion which satisfies
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]
\(||\sigma| > 0\) is not of type (b) then there must be a ground term \( \tau \) which is not reducible in terms of \( x_\mu \), i.e. \( \tau \) cannot satisfy \( \tau = \sigma' \theta(x_\mu) \) for some non-empty prefix \( \sigma' \), and
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx \tau \wedge \tau \approx \theta(x_\mu)). \]

(e) For a recursion with
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]
which is not of type (c) we get for \( x_\mu \) the same property as for \( x_{\mu, k} \) in (d).

(f) If the previous cases do not apply, the recursion might satisfy
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k} \approx \sigma' x_{\mu, k}). \]
with \(|\sigma| > 0\) and \(|\sigma'| > 0\). Since \( x_\mu \) is not eliminable in terms of \( x_{\mu, k} \) and vice versa, there must be ground terms \( \tau, \tau' \) such that
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx \tau \wedge x_{\mu, k} \approx \tau' \wedge \tau \approx \tau'). \]

(g) If a recursion is not of type (a-f) then
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k}). \]
But then \( x_\mu \) and \( x_{\mu, k} \) must be ground eliminable as in (f).

Since a recursion of type (a-c) is not redundant if it contains terms \( \tau \) or \( \tau' \) such that \( \tau \) is not reducible in terms of \( x_\mu \) and \( \tau' \) is not reducible in terms of \( x_{\mu, k} \) and \( \tau \) or \( \tau' \) are used to eliminate \( x_\mu \) and \( x_{\mu, k} \), there must be at least one recursion \( \pi_{m+1} \ldots \pi_{m+k} \) such that
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset (S_{m+k}^x \cup S_{m+1}^x \cup \ldots \cup S_{m+k}^x) \}
and \( \theta \) still follows either by ground inferences or due to the properties of (b) and (c).

If \( l_{max} = \max(|w| | (A, w), S_r \in R) \) then the following pumping lemma follows immediately as a corollary.

22. COROLLARY. Suppose that \( w \in V_F^* \) is derivable with \( \phi \equiv \exists x_1 \ldots x_k' (\wedge S^x (x_1, \ldots, x_k') \) over \( \pi \) of length \( n \), that \( \exists x_1 \ldots x_k' \) \( S (x_1, \ldots, x_n) \) is a normal form of \( \phi \) and that \( M_F = (\langle \delta', \wp \rangle, \wp) \) is a minimal model of \( \phi \).
[PROOF.

If \( |w| > |w|_{\text{m}} \) then at least one path of \( w_F \) is longer than \( |\delta'| \) and more than \( 2|\delta'| + 1 \) different nodes on that path are labelled by the same \( A \in V_N \). Without loss of generality we assume again that \( \pi \) is a derivation whose c-structure derivation \( \omega \) has the form \( \omega_m \ldots \omega_{m_1} \omega_m \) with \( \omega_m = s[t, A_m, y]_t \) for each node \( m \) on that path which is labelled by \( A \), and that \( \omega \) is non-deterministically constructed from \( S^x \) by recursive variable substitution, i.e. we require for each \( (x, \sigma, \tau) \) either \( x \approx \tau \in S^x \) or \( \exists x \approx \tau \in S^x (y, \tau) \in \theta \). Suppose furthermore that \( S \subseteq S^x \) with \( S = S[\theta] \). In order to isolate the recursion which allows pumping we have to distinguish the following cases.

1. If \( \pi \) contains a recursion \( \pi_{m+1} \ldots \pi_{m+k} \) with \( \omega_m = s[t, A_m, y]_t \), \( \omega_{m+k} = s[t, A_m, y]_t \) and \( |w| > 0 \) and \( x_\mu \) and \( x_{\mu, k} \) do not occur in \( S^x \), we take \( \pi_{m+1} \ldots \pi_{m+k} \).

2. If \( \pi \) does not contain such a recursion there must be at least three distinct A-labelled nodes \( \iota, \lambda, \iota, \lambda, \nu, \lambda \) on that path such that
\[ \omega_\iota = s[u', A_\iota, y]_t, \]
\[ \omega_{\iota+l} = s[u', A_\iota, x'_{\iota+l}, y]_t, \]
\[ \omega_{\iota+l+u} = s[u', A_\iota, y], A_{\iota+l}, \lambda, x'_{\iota+l}, y]_t \]
with \(|u'| > 0\) and \(|\sigma| > 0\) and
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k} \approx x_{\mu, k}). \]

2.1 Suppose there is a recursion of type (a-c) (cf. proof of lemma 21) we choose this one.

2.2 If \( \pi \) does not contain such a recursion each of the recursions must be of type (d), (e), (f) or (g).

But then there must be one recursion \( \pi_{m+1} \ldots \pi_{m+k} \) (\( m = i, k = l \) or \( m = i + l, k = v \)) with
\[ \exists x_1 \ldots x_k' (\wedge S^x \supset x_\mu \approx x_{\mu, k} \approx x_{\mu, k}). \]
for some ground terms \( r, r' \). This recursion is chosen for the proof.

On the basis of the recursion \( \pi_{m+1} \rightarrow \pi_{m+1} \) we can now define derivations \( \pi^i \) as follows. We set \( \pi^1 = \pi \) and define \( \pi^{i+1} \) on the basis of \( \pi^i \) by

\[
\pi^{i+1}_j = \begin{cases} 
(\{ s[u, A[v^i r x^i]_j, k, i]\}, y, S) & \text{if } j > m + ik \\
\pi^{i+1}_{j-k} = (\{ s[u, A[v^i r x^i]_{j-k}, k, i]\}, y, S) & \text{if } j \leq m + ik 
\end{cases}
\]

and

\[
\omega^{i+1}_j = \{ a_i \} \quad \text{if } j \leq m + ik
\]

By induction on \( i \) it can then be shown for all possible cases that \( \therefore \exists x_1 \ldots x_k. (\prod S^x \equiv \prod S^y) \). \( \Box \)

### 3.2 Invariance of the Parameters under Deductive Equivalence

Since the universes of the minimal models of two deductively equivalent consistent feature descriptions must have the same cardinality, for the completion of the proof of theorem 13 it remains to be shown that two deductively equivalent consistent feature descriptions have the same "information content", i.e. that the sets of literals of their normal forms have the same cardinality:

**23. Lemma.** Suppose that \( \exists x_1 \ldots x_l \land S_v [x_1 \ldots x_l] \) and \( \exists y_1 \ldots y_k \land S_v [y_1 \ldots y_k] \) are deductively equivalent consistent feature descriptions in normal form then \( |S_v [x_1 \ldots x_l]| = |S_v [y_1 \ldots y_k]| \). We prove lemma 23 in two steps. First, we show that we can convert \( S'_{v} [y_1 \ldots y_k] \) into a set \( S''_{v} [x_1 \ldots x_l] \) with the same cardinality such that \( \exists x_1 \ldots x_l (\land S_v [x_1 \ldots x_l] \equiv \land S''_{v} [x_1 \ldots x_l]) \) holds.

**24. Lemma.** Assume that \( \exists x_1 \ldots x_l \land S_v [x_1 \ldots x_l] \) and \( \exists y_1 \ldots y_k \land S_v [y_1 \ldots y_k] \) are deductively equivalent consistent feature descriptions in normal form. Then \( l = k \) and there is a set \( S'_{v} [x_1 \ldots x_l] \) with

(i) \( |S'_{v} [x_1 \ldots x_l]| = |S''_{v} [y_1 \ldots y_k]| \) and

(ii) \( \exists x_1 \ldots x_l (\land S_v [x_1 \ldots x_l] \equiv \land S''_{v} [x_1 \ldots x_l]) \).

**Proof.** Suppose that \( \{ x_1 \ldots x_l \} \cap \{ y_1 \ldots y_k \} = \emptyset \) (rename if necessary), that \( S''_{v} [y_1 \ldots y_k] \) is in reduced form (the reduction of an independent set does not change the cardinality) and that \( M = (U, \exists) \) is an arbitrary model of \( \exists x_1 \ldots x_l \land S_v [x_1 \ldots x_l] \) and \( \exists y_1 \ldots y_k \land S_v [y_1 \ldots y_k] \). Let \( \alpha \) and \( \alpha' \) be assignments such that \( \alpha = M \models S_v [x_1 \ldots x_l] \) and \( \alpha' = M \models S_v [y_1 \ldots y_k] \).

We show first that there is a bijective function \( \theta : \{ x_1 \ldots x_l \} \rightarrow \{ y_1 \ldots y_k \} \) such that for all \( x_i \) there is a \( x_i' \in F^\pi \) (\( i = 1 \ldots l \)) and a variable \( \theta(x_i) \) occurring in \( S' \) with \( \alpha(x_i) = \exists (\theta(x_i))(\alpha) \). First of all \( \forall \theta \) is left-total, since \( \alpha(x_i) = \exists (\tau) \) with \( \tau \) variable-free would imply that \( x_i \) is eliminable in \( S' \). In order to show that \( \theta \) is a function, assume \( \alpha(x_i) = \exists (\theta(x_i))(\alpha') = \exists (\theta(y_i))(\alpha') \) for \( y_i, y_h \) occurring in \( S' \) with \( y_i \neq y_h \). Since \( y_i \) and \( y_h \) are not eliminable in \( S' \) there must be terms \( \tau_j, \tau_h \) such that \( \alpha'(y_j) = \exists (\tau_j)(\alpha) \), \( \alpha'(y_h) = \exists (\tau_h)(\alpha) \) and there is no \( \sigma \in F^\pi \) such that \( \exists (\sigma)(\alpha) = \exists (\tau_j)(\alpha) \) or \( \exists (\sigma)(\alpha) = \exists (\tau_h)(\alpha) \). Thus, \( \tau_j \) and \( \tau_h \) must be terms in \( T(x_1 \ldots x_l) \{ x_i \} \) and \( x_i \) would be eliminable in \( S' \). Suppose now that \( y_i \) is not in the range of \( \theta \). Then there must be a term \( \tau \) with \( \exists (\tau)(\alpha) = \alpha'(y_i) \). Since \( y_i \) is not eliminable in \( S' \), \( \tau \) must be of the form \( \sigma x_i \) and there must be a term \( \tau' \) with \( \alpha(x_i) = \exists (\tau')(\alpha') \). If \( \tau' \) is a term in \( T(y_1 \ldots, y_k) \{ y_j \} \), \( y_j \) would be eliminable. Otherwise \( \tau' \) is of the form \( \sigma y_j \) and we would get \( \alpha(x_i) = \exists (\sigma y_j)(\alpha') \). Hence \( \theta \) is onto. Assume finally \( \alpha(x_i) = \exists (\sigma x_i)(\alpha') \) and \( \alpha(x_i) = \exists (\sigma y_i)(\alpha') \) with \( x_i \neq x_j \). Then there must be a term \( \tau \) with \( \alpha(y_i) = \exists (\tau)(\alpha) \). Since \( x_i \) and \( x_j \) would be eliminable in \( S' \) if \( \tau \) is a term in \( T(x_1 \ldots, x_l) \{ x_i, x_j \} \), \( \tau \) is of the form \( x_i x_j \) or \( x_j x_i \). But then either \( \alpha(x_i) = \exists (\sigma x_i)(\alpha') \) or \( \alpha(x_i) = \exists (\sigma y_i)(\alpha') \). Thus, \( \exists \) is bijective and \( l = k \).

On the basis of \( \theta \) we then define a sequence of new sets \( S'_i (0 \leq i \leq l) \) by induction as follows (within the induction we assume \( \exists (x_1) = y \)):

\[
S'_0 = S'_v [y_1 \ldots y_k] \quad \text{if } \alpha(x_i) = \alpha'(y)
\]

\[
S'_i = \{ S'_{i-1} \[ y \equiv \exists \sigma(y') ] \} \cup \{ x_i \equiv \sigma x_i \} \quad \text{if } (A).
\]

where \( (A) \) means \( \alpha(x_i) \neq \alpha'(y), y \equiv \exists \sigma(y') \in S'_{i-1} \) and \( \alpha(x_i) = \exists (\sigma y_i)(\alpha') \). In the case where the variables refer to different nodes on a loop (\( \alpha(x_i) \neq \alpha'(y) \)) the definition is well-formed, since \( S'_i [y_1 \ldots y_k] \) is reduced and normalized and thus there must be exactly one equation \( y \equiv \exists \sigma(y') \) in \( S'_{i-1} \) describing the loop with the node to which \( x_i \) refers. For \( S''_i [y_1 \ldots y_k] \), \( |S''_i [x_1 \ldots x_l]| = |S'_i [y_1 \ldots y_k]| \) follows immediately by induction on the construction of \( S''_i \). Finally we get (ii), since

\[
\exists x_1 \ldots x_l (\exists x_{i+1} \ldots x_l \land S_v [x_1 \ldots x_l] \equiv \exists \theta(x_{i+1}) \ldots \theta(x_l) \land S'_i [x_1 \ldots x_l], \theta(x_{i+1}), \ldots, \theta(x_l))
\]

can easily be verified by induction on the construction of \( S''_i \). \( \Box \)

Since two deductively equivalent independent and consistent sets of (variable-free) literals reduce to the same set of literals in reduced form, lemma 25 follows by skolemization and completes the proof of lemma 23.

**25. Lemma.** If \( \exists x_1 \ldots x_l \land S_v [x_1 \ldots x_l] \) is a consistent feature description in normal form then \( \exists x_1 \ldots x_l (\land S_v [x_1 \ldots x_l] \equiv \land S' [x_1 \ldots x_l]) \). Finally we get (ii), since

Appendix: Syntax and Semantics of Feature-Description Languages

A feature-description language \( L(C) \) consists of the logical connectives \( \sim \) (negation), \( \lor \) (implication), the equality symbol \( = \), the existential quantifier \( \exists \) and the parentheses \( (\ldots) \). The nonlogical vocabulary is given by a finite set of constants \( V \) (atomic values),
a possibly empty finite set of constants $C$ (Skolem-constants) and a finite set of unary partial function symbols $F_1$ ($V, C, F_1$ pairwise disjoint). The class of terms and formulas of $L(C)$ are recursively defined as usual. Feature descriptions of $L(C)$ are expressions of the form $\exists x_1 . . . x_l \land S[x_1, . . . , x_l]$, where $S$ is a finite set of (quantifier-free) literals. (We assume that the connectives \lor (disjunction), \land (conjunction) and \equiv (equivalence) are introduced by their usual definitions.)

A model for $L(C)$ consists of a nonempty universe $\mathcal{U}$ and an interpretation function $\mathfrak{I}$. Since not every term denotes an element in $\mathcal{U}$ if the function symbols are interpreted as unary partial functions, we generalize the partiality of the denotation by assuming that $\mathfrak{I}$ itself is a partial function. It is only required that all Skolem-constants denote. Suppose $[X \rightarrow Y]$ designates the set of all partial functions from $X$ to $Y$ and $[X \rightarrow Y]$ the set of all total functions from $X$ to $Y$, then a model is defined as follows: \footnote{If constant-consistency and constant/complex-consistency are to be guaranteed for the atomic values $V$ or acyclicity has to be ensured for the models, special conditions can be added to the definition.}

**Definition.** A model for $L(C)$ is a pair $M = (\mathcal{U}, \mathfrak{I})$, consisting of a nonempty set $\mathcal{U}$ and an interpretation function $\mathfrak{I} = \mathfrak{I}_V \cup \mathfrak{I}_C \cup \mathfrak{I}_{F_1}$, such that

1. $\mathfrak{I}_V \subseteq [V \rightarrow \mathcal{U}]$,
2. $\mathfrak{I}_C \subseteq [C \rightarrow \mathcal{U}]$,
3. $\mathfrak{I}_{F_1} \subseteq [F_1 \rightarrow [\mathcal{U} \rightarrow \mathcal{U}]]$,
4. $\forall f \in F_1 (f \in \text{Dom}(\mathfrak{I}) \rightarrow \mathfrak{I}(f) \neq \emptyset)$.

If we extend the denotation function to terms and variable assignments $\alpha$, the definition of the satisfaction relation differs only in the clause for the equations from the usual one:

$\alpha \models_M \tau \equiv \tau'$ iff $\mathfrak{I}(\tau)(\alpha)$ and $\mathfrak{I}(\tau')(\alpha)$ are defined and $\mathfrak{I}(\tau)(\alpha) = \mathfrak{I}(\tau')(\alpha)$.

**References**

Beierle, C., and U. Pletat. 1988. Feature Graphs and Abstract Data Types: A Unifying Approach. In Proceedings of the 12th International Conference on Computational Linguistics. Budapest.

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

Johnson, M. 1988. Attribute-Value Logic and the Theory of Grammar. Chicago: CSLI Lecture Notes Series, Chicago University Press.

Knuth, D., and P. Bendix. 1970. Simple Word Problems in Universal Algebra. In J. Leech, ed., Computational Problems in Universal Algebra. Elmsford: Pergamon Press.

Shieber, S., H. Uszkoreit, F. Pereira, J. Robinson, and M. Tyson. 1983. The Formalism and Implementation of PATR-II. In B. Groz and M. Stickel, eds., Research on Interactive Acquisition and Use of Knowledge. SRI Final Report 1984. SRI International, Menlo Park.

Statman, R. 1977. Herbrand’s Theorem and Gentzen’s Notion of a Direct Proof. In J. Barwise, ed., Handbook of Mathematical Logic. Amsterdam: Elsevier North-Holland.

Wedekind, J. 1991. Classical Logics for Attribute-Value Languages. In Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics. Berlin.

Wedekind, J. 1994. Some Remarks on the Logic of Unification Grammars. In C. J. Rupp, M. Rosner, and R. Johnson, eds., Constraints, Language and Computation. London: Academic Press.