Using Constraints in a Constructive Version of GPSG

Wilhelm WEISWEBER
Technical University of Berlin
Institute for Software and Theoretical Computer Science
Project Group KIT
Sekr. FR 5-12
Franklinstr. 28/29
D-1000 Berlin 10
Email: weisweb@db0tui11.bitnet

Abstract

Complex categories are caracteristic of unification grammars as for example GPSG [Shieber86a]. They are sets of pairs of features and values. The unification, which can be applied to two or more categories, is the essential operation.

The papers of [Shieber85], [Barton85] and [Ristad86] deal with the influence of complex categories on the efficiency of the parsing algorithm. This is one problem from using complex categories, another one arises when using a constructive version of GPSG (see [Busemann/Hauenschild88] in this volume). Namely that the application of admissibility conditions, e.g. LP statements and FCRs [1], to a local tree t is prevented because particular feature values of categories in t are not yet specified, but they will be instantiated later somewhere else in the complete tree. Similar problems are described in [Karttunen86] for D-PATR.

This work describes the latter problem and will present a solution working with computation, evaluation and propagation of constraints within local trees (depth 1). The constraint evaluation will reject local trees if the constraints of the subtrees of the daughters are violated.

1 Introduction

First of all some fundamentals of a constructive version of GPSG which has been developed within the projects KIT-NASEV and its successor KIT-FAST [2] will be described (for details see [Hauenschild/Busemann88], [Keller87], [Busemann87] and [Weisweber87]). The reader's familarity with the fundamental knowledge of GPSG as presented in [GKPS85] will be assumed.

1.1 The Grammar Format

The ID/LP format of the grammar allows the explicit formulation of generalizations about the partial order of the daughters of the ID rules via LP statements. ID rules are tuples of the form (⟨mother⟩ → ⟨daughters⟩), for example (S → {NP, VP}), where ⟨daughters⟩ is a multi-set of categories which can be dominated by the category ⟨mother⟩. Lexical rules are of the form (⟨mother⟩ → ⟨wordform⟩). LP statements are of the form '$\langle \text{category}_1 \rangle < \langle \text{category}_2 \rangle$', for example the LP statement NP < VP requires that the category NP must precede the category VP in a sequence of daughters of any local tree licensed by an ID rule of the grammar.

For every grammar a set $F = \{f_1, f_2, ..., f_n\}$ of syntactic features exists with $|F| = n$. The number 'n' of features can vary from grammar to grammar, but for a particular grammar it is constant. Each feature f_i has its domain $D(f_i)$. A complex category C is an n-tuple with $C \in D(f_1) \times D(f_2) \times ... \times D(f_n)$ and the position 'i' of the tuple represents the value of the feature f_i.

Example : F = {N, V, BAR, PER, PLU, GEN}

position i	feature f_i	domain $D(f_i)$
1	N	{+,-}
2	V	{+,-}
3	BAR	{0,1,2}
4	PER	{1,2,3}
5	PLU	{+,-}
6	CAS	{nom,gen,dat,acc}

The category C = [+N,-V,BAR 2,nom] will be represented as (+,-,2,X,_,nom). C is the category traditionally called a nominative NP. If a feature value of a category is not specified, like PER and PLU in C, it will be noted as a variable [3]. The value of a feature f_i of

[1] LP = Linear Precedence; FCR = Feature Co-occurence Restriction

[2] KIT-FAST (FAST = Functor-Argument Structure for Translation; KIT = Künstliche Intelligenz und Textverstehen = Artificial Intelligence and Text Understanding) as well as its predecessor KIT-NASEV (NASEV = Neue Analyse- und Syntheseverfahren zur maschinellen Übersetzung = New Methods of Analysis and Synthesis for Machine Translation) constitute the Berlin component of the complementary research project of EUROTRA-D, which receives grants by the Federal Minister for Research and Technology under contract 1013211.

[3] A variable feature value will be noted as a capital letter (e.g. X, Y, Z) when the same variable value is needed at another place, otherwise it will be noted as '_

the category C can be expressed with $C(f_i)$, e.g. C(CAS) = nom or C(PER) = X. It is possible that a feature can have a category as its value, e.g. SLASH.

In the following some predicates on feature values are used. They are 'spec', 'atom' and 'cat'. The semantics of them is as follows:

spec(C(f)) $\Leftrightarrow$ C(f) is specified either with an atom or a category
atom(C(f)) $\Leftrightarrow$ C(f) is specified with an atom
cat(C(f)) $\Leftrightarrow$ C(f) is specified with a category

1.2 Feature Instantiation

The Feature Instantiation Principles [4] are not the subject of this paper. Only one thing needs to be said at this point, that is that the FIPs are not only regarded as filters for local trees as in [GKPS85], but in the constructive version of GPSG they propagate values of features within a local tree from the mother to the daughters and vice versa, or from one daughter to another if this feature value is affected by one of the FIPs (i.e. it is a HEAD, FOOT or agreement feature; a brief outline of the work which is done by the AP is given in section 2.1). In this case the FIPs are construction principles which propagate information from one point of a local tree to possibly many other points which require this information. In cases where a value is not specified (i.e. it is a variable), the variable is propagated. The FIPs are also filters for local trees and will reject local trees which have categories where the values of one of the features are not consistent.

Before discussing the Feature Co-occurrence Restrictions (FCRs) two definitions are necessary to be able to define legal categories.

Definition*: *extension

A category E is the extension of a category C ($C \sqsubseteq E$) iff
(i) $\forall f \in F$: atom(C(f)) $\Rightarrow$ atom(E(f)) $\wedge$ C(f) = E(f) [5] and
(ii) $\forall f \in F$: cat(C(f)) $\Rightarrow$ cat(E(f)) $\wedge$ $C(f) \sqsubseteq E(f)$

Definition*: *unifiable

Two categories A and B are unifiable ($A \sqcup B$) $\Leftrightarrow$
$\forall f \in F$: $\neg$spec(A(f)) $\vee$ $\neg$spec(B(f)) $\vee$
((atom(A(f)) $\wedge$ atom(B(f))) $\Rightarrow$ (A(f) = B(f))) $\vee$
((cat(A(f)) $\wedge$ cat(B(f))) $\Rightarrow$ ($A(f) \sqcup B(f)$))

The FCRs in the constructive version of GPSG are not only predicates on categories, they are also modified to become more functional by instantiating variable feature values if necessary. FCRs are implications of the form (n, $A \supset B$). 'n' is the number of this FCR, 'A' is the condition category for the application of this FCR and 'B' is the consequence category. This FCR is applicable to a category C if C is an extension of A and, if so, C has to be unifiable with B. If C and B are not unifiable the category C is not legal.

Definition*: *legal

A category C is legal $\Leftrightarrow$
$\forall (n, A \supset B) \in$ FCR: $(A \not\sqsubseteq C) \vee (A \sqsubseteq C \wedge B \sqcup C)$

If the FCRs are applied to the category C and an FCR (n, $A \supset B$) is applicable to C and the consequence category B is unifiable with C and at least one feature f exists where $\neg$spec(C(f)) and spec(B(f)) ($B \not\sqsubseteq C$), then all those values C(f) are instantiated with B(f) (C(f) := B(f)). The FCRs have to be applied to a category C until no feature values of C are instantiated by them any longer, because the instantiation may cause other FCRs to be applicable.

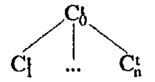
1.3 The Admissibility of Trees

To generate syntactic structures (trees) during analysis or synthesis, ID rules are mapped into local trees in which all categories are legal. This mapping is called a projection.

Definition*: *projection

The projection $\Phi \subseteq ID \times LT$ is a relation from the set ID of ID rules to the set LT of local trees. A tuple (r,t) is an element of Φ ($(r,t) \in \Phi$) where $r = (r', C_0 \rightarrow \{C_1, ..., C_n\})$ and

t =



iff it meets the condition $\phi(C_i) = C^t_j$ with $0 \leq i,j \leq n$ where the total one-to-one, onto function ϕ maps the set $\{C_0, ..., C_n\}$ of categories of the ID rule r' into the set $\{C^t_0, ..., C^t_n\}$ of categories of the local tree t:

$$\phi: \{C_0, ..., C_n\} \rightarrow \{C^t_0, ..., C^t_n\}$$

The function ϕ meets the following conditions:
$\phi(C_0) = C^t_0 \wedge \phi(C_i) = C^t_j \wedge 1 \leq i,j \leq n \wedge$
$\forall \phi(C_k): 0 \leq k \leq n \wedge \phi(C_k)$ is legal $\wedge$ $(C_k \sqsubseteq \phi(C_k))$

When a local tree has been proved to be a projection of an ID rule, the FIPs are applied to it. Despite the fact that the projection already includes the application of the FCRs to the categories of a local tree, they still have to be applied everytime one of the FIPs (FFP, AP, HFC) has been applied to the local tree, because each of them may instantiate a feature value of a category of the local tree and therefore another FCR may be applicable to that category. The last check for admissibility is the application of the LP statements. The sequence of the daughters of the local tree must not conflict with any LP statement of the grammar (it has to be LP-consistent). Before 'LP-consistency' can be defined, the transitive LP-relation (LP^+) has to be defined.

Definition*: *transitive LP-relation

1) $(C_1 < C_2) \in LP \Rightarrow (C_1 \stackrel{+}{<} C_2) \in LP^+$

2) $(C_1 \stackrel{+}{<} C_2) \in LP^+ \wedge (C'_2 \stackrel{+}{<} C_3) \in LP^+ \wedge C'_2 \sqsubseteq C_2 \Rightarrow$
$(C_1 \stackrel{+}{<} C_3) \in LP^+$

[4] In the following they are called FIPs. The FIPs are the Foot Feature Principle (FFP), the Agreement Principle (AP) and the Head Feature Convention (HFC). For detailed discussion of the FIPs see [Busemann87], [Hauenschild/Busemann88] and [Weisweber87].

[5] '$\wedge$', '$\vee$', '$\neg$', '$\Leftrightarrow$' and '$\Rightarrow$' are the logical operators 'and', 'or', 'not', 'equivalence' and 'implication', respectively.

With the help of this definition we are able to define LP-consistency.

Definition*: *LP-consistency

A local tree t is LP-consistent with respect to a grammar with the transitive LP-relation $\not<$ iff

$\forall C_i^t, C_j^t: 1 \leq i,j \leq n \wedge C_i^t$ precedes $C_j^t \Rightarrow$

$\neg\exists C_i', C_j': C_i' \sqsubseteq C_i^t \wedge C_j' \sqsubseteq C_j^t \wedge (C_j' \not< C_i') \in LP^+$

In other words, when a category C_i^t precedes a category C_j^t in a local tree t, they must not conflict with the transitive LP-relation (C_i^t and C_j^t must not be extensions of two categories C_i' and C_j', respectively, where C_j' must precede C_i').
Now the definition of the admissibility of trees can be given.

Definition*: *admissibility of trees

A tree is admissible iff all of its local trees are admissible. A local tree is admissible iff it is a projection of a lexical rule or iff it
- is the projection of an ID rule and
- satisfies all of the FIPs and
- is LP-consistent.

2 LP-Consistency and Legality

The first chapter of this paper illustrates how to build up the syntactic structure (a tree) in the constructive version of GPSG, but it also sketches roughly the way this can generally be done in unification grammars. First of all complex categories are assembled to form a local tree and subsequently the feature values of the categories are instantiated by different procedures (for example in GPSG by the FIPs and in the Lexical Functional Grammar (LFG) described in [Bresnan/Kaplan82] by the evaluation of the f-descriptions). The admissibility of those local trees is determined by some predicates (in GPSG by the LP statements and the FCRs and in LFG by existential constraints with the operator '$=_c$' and the negative existential constraints with the special value 'none'). These predicates can only be applied when particular feature values are specified. But when the admissibility of a local tree is to be proved, it cannot be guaranteed that all feature values have been locally instantiated. In some cases it is possible that a feature value is not locally instantiated rather that it is instantiated somewhere else in another local tree belonging to the complete tree and therefore the admissibility of a local tree is not a local matter anymore.
In this chapter the problems which arise from checking the LP-consistency and the legality in the constructive version of GPSG are described in the sections 2.2 and 2.3 respectively. Section 2.4 briefly outlines the two possible solutions proposed by [Keller87]. But first of all a sample grammar is given to be used for examples.

2.1 A Sample Grammar

To illustrate the problems arising from checking the LP-consistency of a sequence of complex daughter categories and from checking the legality of complex categories of a local tree, a sample grammar is given here.

The set of syntactic features is $F = \{f_1, f_2, f_3\}$. A category is a triple $C \in D(f_1) \times D(f_2) \times D(f_3)$.

Features:	f_1	$D(f_1) = \{a,b,c,d,e\}$
	f_2	$D(f_2) = \{+,-,*\}$
	f_3	$D(f_3) = \{1,2,3\}$

$$\begin{array}{lll} ID = \{ & ((a,X,1) \rightarrow \{(b,Y,1), (c,Z,2)\}), & (1) \\ & ((b,X,\_) \rightarrow \{(d,Y,3), (e,*,1)\}) \} & (2) \end{array}$$

$$\begin{array}{ll} \text{Lexical rules: } \{ & ((e,\_,\_) \rightarrow e), ((d,\_,\_) \rightarrow d), \\ & ((c,-,\_) \rightarrow cm), ((c,+,\_) \rightarrow cp) \} \end{array}$$

$$\begin{array}{llll} LP = \{ (\_,*,\_) < (\_,+,\_), & LP^+ = \{ & (\_,*,\_) \not< (\_,+,\_), & (1) \\ \quad (\_,-,\_) < (\_,*,\_) \} & & (\_,-,\_) \not< (\_,*,\_), & (2) \\ & & (\_,-,\_) \not< (\_,+,\_) \} & (3) \end{array}$$

$$FCR = \{ (1, (b,-,\_) \supset (\_,\_,1)), (2, (d,+,\_) \supset (\_,\_,2)) \}$$

Suppose that the feature f_2 is an agreement feature and that a local tree t which is a projection of this ID rule has been constructed, then the Agreement Principle (AP) forces $X = Y = Z$ and therefore the AP has to consider three cases [6]:

1) If at least two values are instantiated with different values then the AP has to reject t (the predicative view of the FIPs which is still preserved in the constructive version of GPSG).
2) If at least one value is instantiated then the other variable values are instantiated with that value by the AP (propagation of instantiated feature values).
3) If all values are not specified, i.e. they are variables, then the AP will identify all values with one variable (propagation of variable feature values).

Whenever an admissible local tree t is a projection of ID rule (1), the values of the feature f_2 (X, Y, Z) have to be identical and we can apply case (3) to the local tree t.

2.2 ID/LP Specific Problems

In this section only LP-consistency is considered and the legality of categories is ignored. In some cases there are categories of local trees which have feature values not yet specified when the LP-consistency has to be checked, and this possibly means that one or more (transitive) LP statements cannot be applied to the given sequence of daughters.

There are two strategies for processing natural language with ID/LP grammars (IDLPG):

1) The indirect method, where an IDLPG is translated into an equivalent context-free grammar (CFG) (see [Kilbury84]).
2) The direct method, where the ID rules and LP statements are used directly during processing (see [Shieber84], [Kilbury84], [Dörre/Momma85], [Busemann87] and [Weisweber87]).

No matter which method is used some problems arise. Firstly the

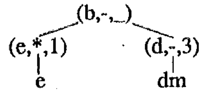
[6] For the sake of simplicity the AP of the constructive version of GPSG is not described in detail here (see footnote 4).

problems in using the indirect method are described. Suppose ID rule (2) is to be translated into equivalent context-free rules. In order to do that, all permutations of the daughters have to be computed and the LP-consistency of the resulting sequences has to be proved. Thus the possible candidates for context-free rules are the following:

(2a) ((b,X,_) → (e,*,1)(d,X,3)) and (2b) ((b,X,_) → (d,X,3)(e,*,1))

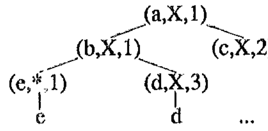
To prove the LP-consistency of (2a), ($C_1 \not\leq C_2$) must not be an element of LP^+, where $C_1 \sqsubseteq$ (d,X,3) and $C_2 \sqsubseteq$ (e,*,1). If such an element exists, the sequence of daughters in (2a) is not LP-consistent and has to be rejected. The only candidate from LP^+ is (2), but it cannot be applied because (_,-,_) $\not\sqsubseteq$ (d,X,3), and so (2a) is a valid context-free rule. But when X is instantiated with '-' later on during processing, the local tree t_1 which is licensed by (2a) has to be rejected because it violates the transitive LP statement (2).

t_1:



The same problem arises when the direct method is used. Suppose the string 'e d cm' is to be analysed. After the terminal symbol 'e' has been read, it is reduced to (e,_,_). This category can be dominated in ID rule (2) because (e,_,_) and (e,*,1) are unifiable. Then 'd' is read and reduced to (d,_,_) which can also be dominated in ID rule (2) and it is unifiable with (d,X,3). Now the daughters of ID rule (2) are complete and before they can be reduced to the mother category (b,X,1), the LP-consistency of the sequence '(e,*,1)(d,X,3)' as above has to be proved. For the above mentioned reason this sequence is LP-consistent and is reduced to (b,X,1). This category can be dominated in ID rule (1) and up to this point the following partial tree t_2 can be constructed.

t_2:



Every local tree in the partial tree t_2 is LP-consistent. Now the terminal symbol 'cm' is read and reduced to (c,-,_). This category is unifiable with (c,X,2) in t_2 and the local tree licensed by a lexical rule can be added to t_2. When the two categories are unified [7], the variable X is instantiated with '-' everywhere in t_2. The result is that the sequence of daughters '(e,*,1)(d,-,3)' is not LP-consistent anymore, because now it violates the transitive LP statement (2). If instead the next input symbol after the string 'e d' is 'cp', no problems with the LP-consistency arise.

2.3 FCR Specific Problems

Similar problems like those with the LP-consistency appear when the FCRs are applied to the categories of a local tree. Suppose that the partial tree t_2 has already been constructed. All of its categories are legal.

Suppose that the next input symbol after the string 'e d' is 'cp'. This terminal symbol is reduced to (c,+,_) which can be unified with (c,X,2) in t_2 and the variable X is instantiated with '+'. Thus all variables X in t_2 have to be replaced by '+' and the category (d,X,3) becomes (d,+,3). Now FCR 2 is applicable because (d,+,3) is an extension of the condition category (d,+,_) ((d,+,_) $\sqsubseteq$ (d,+,3)), but it is not unifiable with the consequence category (_,_,2) (¬((d,+,3) ⊔ (_,_,2))) and thus it is not legal and has to be rejected.

2.4 Two Possible Solutions

According to [Keller87], the problems described in the sections 2.2 and 2.3 can be solved in two ways. One way would be to check the LP-consistency of all local trees and the legality of all categories after the entire tree for the input string has been constructed. The other way would be to restrict the grammar format and/or the FIPs. The disadvantage of the former solution is its inefficency. The checks have to be done in addition after the processing of the input string is terminated because some trees have not already been rejected, although it would have been possible to do so.

The disadvantage of the latter solution is made obvious by two examples. The format of the categories, for example, can be restricted by assuring that the mother category in a local tree is fully specified, i.e. a feature must not have a variable as its value. This would involve a loss of the grammar's descriptive power. Another way would be to restrict the FIPs by assuring that they don't propagate variable feature values, which would involve GPSG losing some of its generality.

3 Constraints

In both cases (LP-consistency and legality) the problems are caused by categories of local trees which are not extensions of the categories in the LP statements or of the condition categories in the FCRs, but which have been unifiable with them. This is the case when a feature of a category of the local tree has a variable as its value and the same feature value is specified in the corresponding category of the LP statements or the FCRs.

This fact means that the LP statements or the FCRs are not locally applicable in some cases, and so the admissibility of the local trees can only be assured with certain constraints which can be fulfilled later on, when the variable is instantiated during processing.

3.1 Computing Constraints from LP

To compute the constraints resulting from the LP statements, the above mentioned cases have to be first identified. Suppose that the sequence 'A B' of categories is to be checked for LP-consistency and an LP statement B' < A' exists where B ⊔ B' and B' $\not\sqsubseteq$ B and A' $\sqsubseteq$ A. This LP statement is not applicable to 'A B' because B' $\not\sqsubseteq$ B, though B' ⊔ B. This means that at least one feature f exists, where spec(B'(f)) and ¬spec(B(f)). Thus the sequence 'A B' is LP-

[7] In the constructive version of GPSG, unification is used for tree formation. In the version of [GKPS85] the root category R of a subtree has to be identical with a daughter category C of a local tree (i.e. R $\sqsubseteq$ C and C $\sqsubseteq$ R).

consistent under the constraint that $B(f_i) \neq B'(f_i)$ for all features f_i with the above mentioned condition, and since all values $B(f_i)$ for those features f_i are variables ($\neg$spec($B(f_i)$)), they must not be instantiated with the values $B'(f_i)$ which are already specified. When the values $B(f_i)$ are instantiated by the FIPs or by the FCRs in the local tree, there is no problem in determining the admissibility. But when the variable feature values $B(f_i)$ are propagated to the mother by the FIPs, they can possibly be instantiated in another local tree and those variable values $B(f_i)$ have to be constrained.

Computing the set of constraints LP_C

When the LP-consistency of a sequence of daughters in a local tree t is checked for every pair of daughters C^t_i, C^t_j, where $1 \leq i < j \leq n$, the set $LP_C(i,j)$ of LP constraints is computed as follows:

1) $\exists\ (C_j \nless C_i) \in LP^+: C_j \sqsubseteq C^t_j \wedge C_i \sqsubseteq C^t_i \Rightarrow$ t is not LP-consistent

2) $\forall\ (C_j \nless C_i) \in LP^+: (C_j \sqcup C^t_j \wedge C_j \not\sqsubseteq C^t_j) \vee (C_i \sqcup C^t_i \wedge C_i \not\sqsubseteq C^t_i) \Rightarrow$
t is LP-consistent with the LP constraints:
$LP_C(i,j) = \{(C^t_j(f), C_j(f)) \mid \text{spec}(C_j(f)) \text{ and } \neg\text{spec}(C^t_j(f))\} \cup$
$\{(C^t_i(f), C_i(f)) \mid \text{spec}(C_i(f)) \text{ and } \neg\text{spec}(C^t_i(f))\}$

3) $\neg\exists\ (C_j \nless C_i) \in LP^+: (C_j \sqcup C^t_j \wedge C_i \sqcup C^t_i) \Rightarrow$
t is LP-consistent with $LP_C(i,j) = \{\}$

$LP_C(i,j)$ is a set of tuples (V_t,V_p) of feature values. V_t is the variable feature value of a category of a local tree. V_p is the specified feature value of the LP statement which will become applicable to the corresponding daughters of the subtree if the values V_t of all tuples in $LP_C(i,j)$ are specified and equal to their corresponding values V_p. In this case the subtree has to be rejected.
The set of all LP constraints $LP_C(0)$ for the local tree in which C^t_0 is the mother category is

$$LP_C(0) = \{LP_C(i,j) \mid 1 \leq i < j \leq n\} \cup \bigcup_{i=1}^{n} \text{eval\_lp}(LP_C(i))$$

where eval_lp($LP_C(i)$) is the evaluation of the LP constraints of the subtree in which the daughter C^t_i is the root category. The set of LP constraints of a projection of a lexical rule is $LP_C(0) = \{\}$, since lexical rules have only one daughter (a wordform).

The evaluation 'eval_lp'

The evaluation 'eval_lp' is either defined or undefined ($\bot$). If it is undefined, the corresponding local tree is rejected, because one of the subtrees of the daughters of that local tree is not LP-consistent.

1) $\text{eval\_lp}(LP_C(i)) = \bot \Leftrightarrow$
$\exists\ C \in LP_C(i): (\forall\ (V_1,V_2) \in C: \text{spec}(V_1) \wedge \text{spec}(V_2) \wedge V_1 = V_2)$

2) $\text{eval\_lp}(LP_C(i)) = LP_C(i)\ -$
$\{C \in LP_C(i) \mid \exists (V_1,V_2) \in C: \text{spec}(V_1) \wedge \text{spec}(V_2) \wedge V_1 \neq V_2\}$

3) $\text{eval\_lp}(LP_C(i)) = \{C \mid (C = C' - M) \wedge C' \in LP_C(i) \wedge C \neq \{\}\}$
where $M = \{(V_1,V_2) \mid \text{spec}(V_1) \wedge \text{spec}(V_2) \wedge V_1 = V_2\}$

The first case (1) means that if one set C of tuples exists in the set $LP_C(i)$ of one daughter of the local tree, where all values of all tuples are specified and equal, then an LP statement will be applicable somewhere in the subtree of this daughter, and it will reject the subtree because some sister categories in the subtree are not LP-consistent anymore.

The second case (2) removes all sets S of tuples from $LP_C(i)$ of one daughter which include one tuple with two different specified values. This means that the LP statement which has caused the computation of S will not be applicable anymore.
The third case (3) removes all tuples with two equally specified values from all sets of tuples in $LP_C(i)$ of one daughter, because they need not be evaluated for a second time.

3.2 Computing Constraints from FCRs

To compute constraints resulting from FCRs, the categories have to be identified which are not an extension of a condition category of an FCR, but unifiable with it. Suppose that the FCRs are to be applied to the category C, and that an FCR (n, $A \supset B$) exists where $A \sqcup C$ and $A \not\sqsubseteq C$. The FCR 'n' is not applicable to C. This may change if a feature value is instantiated. It is the same situation as in section 3.1, but here the computing of constraints is somewhat different, because the application of an FCR to a category may cause that another FCR will become applicable to that category. FCR constraints only have to be computed when $\neg(B \sqcup C)$, or $B \sqcup C$ and $B \not\sqsubseteq C$ (the case $\neg(B \sqcup C)$ means that if the FCR will be applicable to C, it will reject C, and the second case means that if the FCR will become applicable to C, it will instantiate one or more features in C, but the category always remains legal with respect to this FCR). The case $\neg(B \sqcup C)$ is the crucial one, because the legality of the category C can only be assured with the constraint that the set of possibly applicable FCRs, with the above mentioned conditions on the consequence category, still have to be checked for applicability.

Computing the set of constraints FCR_C

For all categories C^t_i where $0 \leq i \leq n$ in a local tree t the set APP(i) of all numbers of FCRs which may still be applicable to C^t_i is computed as follows:

1) $\exists\ (k, A \supset B) \in FCR: A \sqsubseteq C^t_i \wedge \neg(B \sqcup C^t_i) \Rightarrow C^t_i$ not legal

2) $\forall\ (k, A \supset B) \in FCR: A \sqcup C^t_i \wedge A \not\sqsubseteq C^t_i \wedge$
$(\neg(B \sqcup C^t_i) \vee ((B \sqcup C^t_i) \wedge (B \not\sqsubseteq C^t_i))) \Rightarrow C^t_i$ legal and $k \in$ APP(i)

3) $\forall\ (k, A \supset B) \in FCR: \neg(A \sqcup C^t_i) \vee (A \sqsubseteq C^t_i \wedge B \sqcup C^t_i) \Rightarrow$
C^t_i is legal and APP(i) = {}

The set of all FCR constraints $FCR_C(0)$ for the local tree in which C^t_0 is the mother category is

$$FCR_C(0) = \{(C^t_0, \text{APP}(0))\} \cup \bigcup_{i=1}^{n} \text{eval\_fcr}(FCR_C(i))$$

where eval_fcr($FCR_C(i)$) is the evaluation of the FCR constraints from the subtree in which the daughter C^t_i is the root category. $FCR_C(0)$ is a set of tuples (C_i, APP(i)). C_i is a copy of a category of the subtree in which C^t_0 is the root category and APP(i) includes the numbers of all FCRs which may still be applied to C_i if particular feature values of this category are going to be instantiated. The only new set APP in a local tree is computed from the mother, because the evaluation of the FCR constraints on the subtrees of the daughters includes the application of the applicable FCRs to the daughter categories (because they are the root categories of the subtrees). Thus the remaining tuples of the daughters and their subtrees will be computed by the evaluation. The set of FCR constraints on a projection of a lexical rule is $FCR_C(0) = \{(C^C_0, APP(0))\}$.

The evaluation 'eval_fcr'

The evaluation 'eval_fcr' is either defined or undefined ($\perp$). If it is undefined, the corresponding local tree is rejected, because one or more categories of the subtrees of the daughters of that local tree are not legal.

1) $eval\_fcr(FCR_C(i)) = \perp \Leftrightarrow \exists\, (C,APP) \in FCR_C(i)\text{: } k \in APP \wedge$
$(k, A \supset B) \in FCR \wedge A \sqsubseteq C \wedge \neg(B \sqcup C)$

2) $eval\_fcr(FCR_C(i)) = \{(C,M) \mid (M = APP - S) \wedge$
$(C,APP) \in FCR_C(i) \wedge M \neq \{\}\}$ where
$S = \{k \mid (k, A \supset B) \in FCR \wedge (\neg(A \sqcup C) \vee (A \sqsubseteq C \wedge B \sqcup C))\}$

The first case (1) means that if the set $FCR_C(i)$ of a subtree of one daughter C^C_i includes one tuple (C,APP) in which the category C is not legal with respect to the FCR ($k, A \supset B$) where 'k' is in the set APP of numbers of FCRs, then the subtree has to be rejected, because the category C in this subtree is no longer legal.
The second case (2) removes all the numbers 'k' of FCRs from the set APP of all tuples in $FCR_C(i)$, where the FCR ($k, A \supset B$) is no longer applicable to the category C or where it has been successfully applied to the corresponding category.

3.3 Evaluation and Propagation

After a local tree t has been proved to be a projection of an ID rule, all FIPs are applied to t, the FCRs to its mother, and the set APP(0) of the numbers of all FCRs which possibly will be applicable to the mother, is computed. After that the FCR constraints on the subtrees of the daughters are evaluated, which means that all applicable FCRs are applied to the daughters and to all other categories of their subtrees. The remaining FCR constraints from the evaluation, and the FCR constraint (C^C_0, APP(0)) on the mother, will then be combined to form the new FCR constraint set $FCR_C(0)$ on the subtree in which C^C_0 is the root category. The new FCR constraint set is propagated to the mother.
Next the LP-consistency of the daughters has to be checked, and during this check the new LP constraints on the daughters are computed. These constraints are combined with the LP constraints resulting from the evaluation of the LP constraints on the subtrees of the daughters, to form the entire set of LP constraints $LP_C(0)$ on the subtree in which C^C_0 is the root category which is then also propagated to the mother.

4 Conclusion

With this method of constraint computation, evaluation and propagation, a new definition of the admissibility of trees is necessary.

Definition: admissibility of trees

A tree is admissible iff all of its local trees are admissible and the evaluations 'eval_fcr' and 'eval_lp' of constraints of the root category are defined and both are the empty set { }. A local tree t is admissible iff it is a projection of a lexical rule or iff it
- is a projection of an ID rule with the FCR constraint (C^C_0, APP(0)) on the mother C^C_0 and
- satisfies all of the FIPs and
- is LP-consistent with the LP constraints $LP_C(i,j)$ on all daughters C^C_i and C^C_j which are propagated to the mother where $1 \leq i < j \leq n$ and
- the evaluation 'eval_fcr($FCR_C(i)$)' of every daughter C^C_i is defined where $1 \leq i \leq n$ and their results are propagated to the mother and
- the evaluation 'eval_lp($LP_C(i)$)' of every daughter C^C_i is defined where $1 \leq i \leq n$ and their results are propagated to the mother.

The consequence for the root category R of an entire tree of one input string of a natural language will be the fact that all features f_i of R, where $\neg spec(R(f_i))$, and where f_i is needed for the evaluation of the constraints of the tree, have to be instantiated according to their domain $D(f_i)$ because such a tree represents a class of ambiguous solutions. After that the constraints on every tree of this class are evaluated and only the trees where $FCR_C(0)$ and $LP_C(0)$ are defined and their evaluation is { } are admissible.

References

[Barton85] : G.E. Barton Jr.; "The Computational Difficulty of ID/LP Parsing"; 23rd Ann. Meet. of the ACL at University of Chicago; Chicago 1985

[Bresnan/Kaplan82] : J. Bresnan, R.M. Kaplan; "Lexical Functional Grammar: A Formal System of Grammatical Representation"; in: MIT Press Series on Cognitive Theory and Mental Repr. pp.173-281; Cambridge 1982

[Busemann87] : S. Busemann; "Generierung mit GPSG"; in: K. Morik(ed.) Proceedings of the 11th German Workshop on Artifical Intelligence pp.355-364; Springer-Verlag; Geseke 1987

[Busemann/Hauenschild88] : S. Busemann, C. Hauenschild; "A Constructive View of GPSG or How To Make It Work"; 12th International Conference on Computational Linguistics, Budapest 1988

[Dörre/Momma85] : J. Dörre, S. Momma; "Modifikationen des Earley-Algorithmus und ihre Verwendung für ID/LP-Grammatiken"; Manuscript of the Department for Linguistics/ Romanistics at the University of Stuttgart 1985

[GKPS85] : G. Gazdar, E. Klein, G. Pullum, I. Sag; "Generalized Phrase Structure Grammar"; Oxford; Blackwell 1985

[Hauenschild/Busemann88] : C. Hauenschild, S. Busemann; "A Constructive Version of GPSG for Maschine Translation" in Steiner, Schmidt, Zelinsky-Wibbelt (eds.) "From Syntax to Semantics-Insights from Maschine Translation", Frances Pinter

[Karttunen84] : L. Karttunen; "D-PATR: A Development Environment for Unification-Based Grammars"; 11th International Conference on Computational Linguistics, Bonn 1986

[Keller87] : W. Keller; "An Overview of the Project NASEV Parser"; Manuscript, University of Sussex 1987

[Kilbury84] : J. Kilbury; "Earley-basierte Algorithmen für direktes Parsen mit ID/LP-Grammatiken"; KIT-Report 16; TU Berlin 1984

[Ristad86] : E.S. Ristad; "Computational Complexity of Current GPSG Theory"; 24th Ann. Meet. of the ACL at Columbia University; New York 1986

[Shieber84] : S.M. Shieber; "Direct Parsing of ID/LP Grammars"; Linguistics and Philosophy 7 1984 pp.135-154

[Shieber86a] : S.M. Shieber; "An Introduction to Unification-based Approaches to Grammar"; CSLI Lecture Notes Number 4, Ventura Hall, Stanford University, Stanford 1986

[Shieber86b] : S.M. Shieber; "A Simple Reconstruction of GPSG"; Proceedings of the 11th International Conference on Computational Linguistics; Bonn 1986 pp.211-215

[Weisweber87] : W. Weisweber; "Ein Dominanz-Chart-Parser für generalisierte Phrasenstrukturgrammatiken"; KIT-Report 45, TU Berlin