

Evaluation and Documentation of Natural Language Grammars by Feature Propagation Flow Analysis

Michel Gagnon, Ettore Merlo, Dominic Letarte

Department of Computer Engineering,

École Polytechnique de Montréal,

P.O. Box 6079, Downtown Station,

Montreal, Quebec,

H3C 3A7, Canada

{michel.gagnon, ettore.merlo, dominic.letarte}@polymtl.ca

Giuliano Antoniol

Department of Engineering,

University of Sannio,

Research Centre on Software Technology,

Via Traiano - Palazzo ex Poste I-82100

Benevento - ITALY

antoniol@ieee.org

Abstract

In this paper, we address the problem of computing by flow analysis the propagation of feature values in a constraint-based natural language grammar. The method may effectively be used to evaluate and document grammars, following a formal engineering perspective. By identifying all the possible values for features in a grammar rule, together with all the possible origins and destinations of these values, we can document the structure of feature propagation and unification, and also detect structural errors and idiosyncrasies.

1 Introduction

The construction of a large-scale grammar for natural language parsing is a task that requires many person-years of work. Not only must we build the rules to cover all the possible syntactic forms, but we must also achieve a fine tuning of these rules to take into account all the subtleties of the language and the idiosyncratic forms. The task becomes even worse if our objective is a grammar tolerant to mistakes. Generally, this results in a big grammar and a large lexicon, both with many intricate features.

Usually, it is very difficult to identify the impact of a modification in the grammar or the lexicon. With the tools available at this moment, typically, we would use a corpus of texts and check that the analysis of the sentences remains the same after the modification. By using some similarity metrics, the derivation tree returned by the parser is compared with the manually labeled sentence in the corpus, and the results are used to calculate precision and recall values. See (Lin, 1995; Carroll et al., 1998) for influential results using this approach.

This kind of evaluation is essential in the grammar design process, but unfortunately it does not solve all the problems related to this task. Its major shortcoming is that it does not point specifically at the defects in the grammar design. It is not sufficient to evaluate the correctness of the parses, we must also get a clear

idea of the behavior of the parser as a software product. It would be useful to know what other grammar rules or lexical entries could potentially be affected by some change in a grammar rule. It would also be important to know whether a rule is always used as expected. For example, we want to avoid underconstrained rules that could be activated with some unexpected feature values.

Another important drawback in traditional evaluation methods is that they are highly dependent on linguistic theories. This makes difficult the portability to other grammars. By concentrating the effort on the implementation level, that is, the propagation of feature values in a constraint-based formalism with unification, we get not only a complementary evaluation method, but also a technique easily applicable to another grammar theory based on the same formalism. It results in a quantitative method less prone to subjectivity and relying on principles that are easily explained.

As a first step to the elaboration of a tool for calculating impact measures, we propose to adapt a flow analysis that is well known in software engineering. We will argue that using this simple method, we can easily identify problems in the design of a grammar.

In the next section, we present briefly the grammatical formalism that has been used to test our method. The flow analysis itself is formally described in sections 3 and 4. Finally, a small experiment on a grammar for Portuguese is presented and used to show the advantages of our method.

2 Constraint-based Grammars and Unification Sensitive Derivation Graphs

Flow analysis presented in section 3 has been defined on constraint-based grammars with unification, which are now used in the majority of mainstream grammar formalisms, such as Head-

Driven Phrase Structure Grammar (Pollard and Sag, 1994) and Tree-adjoining Grammar (Vijay-Shanker and Joshi, 1988). In our approach, the grammar rules are composed of syntactic symbols with feature structures. To simplify the description, we assume that the value of a feature may be either an atom or a variable. This simplification makes the understanding of the propagation mechanism described in the next section easier, without limiting its theoretical foundations. We define a grammar G as follows:

$$G = (N, FEAT, VARS, ATOMS, RULES, S)$$

$$\begin{aligned} &propagSet : RULES \rightarrow \\ &\rightarrow FEAT \times \{VARS \cup ATOMS\} \end{aligned}$$

$$\begin{aligned} &unifySets : RULES \rightarrow \\ &\rightarrow (FEAT \times \{VARS \cup ATOMS\})^n, (n \in \mathcal{N}^+) \end{aligned} \quad (1)$$

where N is the set of syntactic symbols, $FEAT$ is the set of features, $VARS$ is the set of variables used for unification purposes, $ATOMS$ is the set of atoms that represent possible values for features, $RULES$ is the set of grammar rules, and S is the start symbol.

Some functions are defined on rules. $propagSet$ returns the feature structure of the left-hand side of the rule, and $unifySets$ returns the n-tuple of feature structures that are contained in the right-hand side.

Starting from this grammatical formalism, we construct a *derivation graph*, which essentially represents how feature values may be propagated during the parsing process. A node in this graph corresponds to a feature structure in a grammar rule. An edge associates the left-hand side of rule with every feature structure found on its right-hand side. There are also edges connecting the left-hand side of a rule and every right-hand side of some other rules (or even the same rule) that is unifiable with it. Figure 2 shows the derivation graph of the small portuguese grammar given in Figure 1. The details of this figure are explained further, after the formal definitions of this graph.

To make the feature flow information propagation sensitive to unification, nodes have been defined of two different types, depending on the role they play. A node is of type *LHS* if it represents a feature structure associated to the left-hand side of a rule. Conversely, a node is of type *RHS* if it corresponds to a feature struc-

Grammar rules:

- (1) $S \rightarrow NP \begin{bmatrix} NUM & X \\ PERS & P \end{bmatrix}, VP \begin{bmatrix} NUM & X \\ PERS & P \end{bmatrix}$
- (2) $NP \begin{bmatrix} NUM & X \\ PERS & 3 \end{bmatrix} \rightarrow DET \begin{bmatrix} NUM & X \end{bmatrix}, N \begin{bmatrix} NUM & X \end{bmatrix}$
- (3) $NP \begin{bmatrix} NUM & X \\ PERS & P \end{bmatrix} \rightarrow PRON \begin{bmatrix} NUM & X \\ PERS & P \end{bmatrix}$
- (4) $VP \begin{bmatrix} NUM & X \\ PERS & P \end{bmatrix} \rightarrow V \begin{bmatrix} NUM & X \\ PERS & P \\ VAL & intr \end{bmatrix}$
- (5) $VP \begin{bmatrix} NUM & X \\ PERS & P \end{bmatrix} \rightarrow V \begin{bmatrix} NUM & X \\ PERS & P \\ VAL & trans \end{bmatrix}, NP \begin{bmatrix} \end{bmatrix}$

Lexicon:

- (6) $DET \begin{bmatrix} NUM & sing \end{bmatrix} \rightarrow o \text{ (the)}$
- (7) $DET \begin{bmatrix} NUM & plur \end{bmatrix} \rightarrow os \text{ (the)}$
- (8) $N \begin{bmatrix} NUM & sing \end{bmatrix} \rightarrow menino \text{ (boy)}$
- (9) $N \begin{bmatrix} NUM & plur \end{bmatrix} \rightarrow meninos \text{ (boys)}$
- (10) $PRON \begin{bmatrix} NUM & sing \\ PERS & 1 \end{bmatrix} \rightarrow eu \text{ (I)}$
- (11) $V \begin{bmatrix} NUM & sing \\ VAL & intr \\ PERS & 3 \end{bmatrix} \rightarrow dormiu \text{ (slept)}$

Figure 1: Example of grammar for Portuguese

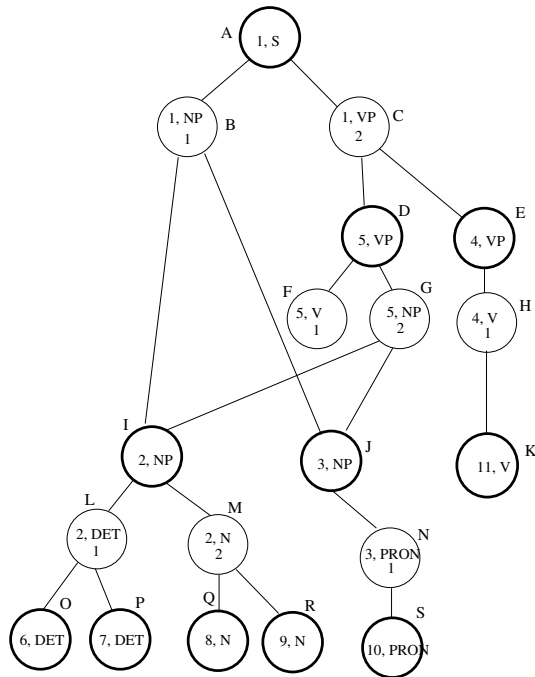


Figure 2: Example of derivation graph

ture from the right-hand side of a rule. Feature flow information from a *LHS* node can be propagated to all *RHS* nodes that are unifiable to it. Flow information from a *RHS* node can only be propagated to the LHS node corresponding to the same rule.

Let us define the function *type*, which returns the type of a node in the derivation graph, the function *rule*, which associates nodes with their corresponding rule, and the function *order*, which associates *RHS* nodes with their positions in the right-hand side of a rule, as follows:

$$\begin{aligned} \text{type} : V_{DG} &\rightarrow \{LHS, RHS\} \\ \text{rule} : V_{DG} &\rightarrow RULES \\ \text{order} : V_{DG} &\rightarrow \mathcal{N} \end{aligned} \quad (2)$$

Functions *type* and *rule* are total functions since they are defined for every node in the derivation graph. Function *order* is a partial function since it is not defined for nodes of type *LHS*.

Let us define $DG = (V_{DG}, E_{DG})$ to be the grammar derivation graph as follows.

- Every rule in the grammar has a corresponding node *LHS* in the graph.
- For every feature structure on the right-hand side of a rule, there is a unique *RHS* node whose order corresponds to its position in the right-hand side of the rule.
- If and only if the left-hand side of a rule is unifiable with a feature structure in the right-hand side of another rule, there is an edge joining their respective nodes in the graph.
- For every two nodes in the graph that correspond to the left-hand side and right-hand side of the same rule, there is an edge joining them.

In the derivation graph showed in Figure 2, bold lines are used to distinguish *LHS* nodes from *RHS* nodes. In each node, the rule number is indicated together with the syntactic symbol associated to this node. For *RHS* nodes, the order is also indicated. Node that the rules corresponding to a lexical entry are represented by a single node in the graph. Beside each node, to simplify the description of flow analysis, a capital letter is used to uniquely identify it.

An additional function *fSet* can be defined on the set of nodes, for ease of later flow analysis description. Function *fSet* returns the feature structure associated to a node in the graph, and is defined as follows:

$$\begin{aligned} fSet : V_{DG} &\rightarrow FEAT \times \{VARS \cup ATOMS\} \\ fSet(v) &= \begin{cases} \text{propagSet}(\text{rule}(v)) & \text{if } \text{type}(v) = LHS \\ [\text{unifySets}(\text{rule}(v))]_{\text{order}(v)} & \text{if } \text{type}(v) = RHS \end{cases} \end{aligned} \quad (3)$$

where $[t]_i$ returns the i^{th} element of the n-tuple t .

3 Flow Equations

Flow analysis statically approximates the dynamic behavior of the unification and derivation processes by computing the fix-point iterative solution of some flow equations.

A general description of flow analysis can be found in (Aho et al., 1986). An approach for constant propagation analysis has been presented in (Merlo et al., 1995). The approach presented in this paper is an improvement of the simple feature value propagation analysis that has been proposed in (Merlo et al., 2004). More precisely, we are now proposing an analysis that takes into account the effect of unification in the propagation of values. Other approaches for flow analysis of logic program can be found in (Debray, 1988; Cousot and Cousot, 1992).

The flow analysis problem can be defined as follows. Let a *def-origin* be a pair (b, w) , where $b \in FEAT$, $w \in V_{DG}$, $\text{type}(w) = LHS$, such that the value of feature b is an atom in the rule $\text{rule}(w)$. In simple terms, a def-origin specifies a rule in which a feature is defined (that is, where its value is an atom) in the left-hand side. At any node v in the graph and for any feature $a \in FEAT$, we may determine the set of def-origins, that is, for any feature in the rule, we may determine the origin of any value that is propagated to this feature.

The flow analysis problem can be modeled by a lattice where each node S_i represents a particular configuration of information about the def-origins for each feature. Suppose there are n distinct features named a_1 to a_n in a grammar G . Formally, a node S_i is denoted by the flow information associated to it as follows:

$$S_i = (V_{i,1}, \dots, V_{i,k}, \dots, V_{i,n}) \quad (4)$$

where $V_{i,k} \subseteq \mathcal{P}(FEAT \times V_{DG})$ is the set of def-origins associated with feature a_k .

The partial order between nodes is defined as follows:

$$S_i \preceq S_j \iff V_{i,k} \subseteq V_{j,k}, 1 \leq k \leq n \quad (5)$$

Equations to compute def-origins for any given grammar rule r are described in terms of the flow information coming out from node $w \in V_{DG}$. The set $OUT(w)$ denotes such information and it also corresponds to one node S_i in the flow problem lattice. We will write $FV(a_m, S_i)$ to denote the set associated to feature a_m in a particular configuration S_i .

Initially, for each node in the derivation graph, flow analysis starts with empty sets for all features, that is, $\forall w \in V_{DG}, OUT(w) = (\emptyset, \dots, \emptyset, \dots, \emptyset)$. Let the flow information coming out from w be $OUT(w) = (V_1, \dots, V_k, \dots, V_n)$. Elements of $OUT(w)$ when $type(w) = LHS$ can be computed as follows:

$$V_k = \begin{cases} \emptyset & \text{if } \nexists (a_k, v) \in fSet(w) \\ \{(a_k, w)\} & \text{if } (a_k, v) \in fSet(w) \wedge \\ & (v \in ATOMS) \\ uSet(x, w) & \text{if } (a_k, x) \in fSet(w) \wedge \\ & (x \in VARS) \end{cases} \quad (6)$$

V_k represents the set of def-origins of the feature a_k propagated by the current node w . If a_k does not belong to the left-hand side of $rule(w)$, i.e., a_k does not belong to the features set of w , it cannot propagate any value, so V_k is the empty set. If the propagation set for $rule(w)$ assigns an atomic value v to a_k , the def-origin of a_k is w itself. Otherwise, V_k is the set of def-origins returned by function $uSet(x, w)$, defined as follows. This set contains every def-origin (b, u) that respects the following conditions:

1. For some node z and feature a_m such that $(z, w) \in E_{DG}$ and $(a_m, x) \in fSet(z)$, $(b, u) \in FV(a_m, OUT(z))$. In other words, there must be in the right-hand side an occurrence of the variable x having (b, u) in its set of def-origins.

2. Let $v = fSet(u)$, the value propagated from node u upto the occurrence of variable x associated to a_m in feature structure $fSet(z)$. In every other occurrence of variable x in the right-hand side of the rule, there is a way to obtain the same value. Put simply, a def-origin will appear on the left-hand side of a rule only if it is found in every occurrence of the same variable on the right-hand side.

Since several feature structures from the LHS of rules may unify with a feature structure on the right-hand side of a rule, we take into consideration the many possible values incoming in a RHS node by simply computing the union of def-origins from the LHS nodes to which it is connected. Let the flow information coming out from RHS node w be $OUT(w) = (V_1, \dots, V_k, \dots, V_n)$ and that coming out from one LHS node z connected to it be $OUT(z) = (V'_1, \dots, V'_k, \dots, V'_n)$. Formally, V_k can be computed as:

$$V_k = \begin{cases} \{(a_k, w)\} & \text{if } (a_k, v) \in fSet(w) \wedge \\ & (v \in ATOMS) \\ \bigcup_{\substack{V'_k \in OUT(z) \\ (z, w) \in E_{DG}}} V'_k & \text{otherwise} \end{cases} \quad (7)$$

It can be shown that flow equations 6 and 7 preserve the lattice partial order, because, if $OUT(z)$, z predecessor of w , becomes bigger in the partial order sense, then $OUT(w)$ also becomes bigger. The convergence of the fix-point iterative solution of flow equations is therefore guaranteed to converge.

Let us now return to our grammar example. A lattice node in the flow analysis will contain three elements, one for each feature in the grammar. We will consider that the first, second and third elements correspond to the three features NUM, PERS and VAL, respectively. After the first iteration, the only information we have is the set of features that are defined explicitly in a feature structure. For example, node I, which corresponds to the left-hand side of Rule 2, will have only one origin specified for the value of feature PERS, which is the same feature structure (because the second item of the pair is the node itself). After a few iterations, we get at some final state where no more change occurs

between two iterations (see Table 1).

w	$OUT(w)$
A	$(\emptyset, \emptyset, \emptyset)$
B	$(\{(NUM, O), (NUM, P), (NUM, Q), (NUM, R), (NUM, S)\}, \{(PERS, I), (PERS, S)\}, \emptyset)$
C	$(\{(NUM, K)\}, \{(PERS, K)\}, \emptyset)$
D	$(\emptyset, \emptyset, \emptyset)$
E	$(\{(NUM, K)\}, \{(PERS, K)\}, \emptyset)$
F	$(\emptyset, \emptyset, \{(VAL, F)\})$
G	$(\{(NUM, O), (NUM, P), (NUM, Q), (NUM, R), (NUM, S)\}, \{(PERS, I), (PERS, S)\}, \emptyset)$
H	$(\{(NUM, K)\}, \{(PERS, K)\}, \{(VAL, H)\})$
I	$(\{(NUM, O), (NUM, P), (NUM, Q), (NUM, R)\}, \{(PERS, I)\}, \emptyset)$
J	$(\{(NUM, S)\}, \{(PERS, S)\}, \emptyset)$
K	$(\{(NUM, K)\}, \{(PERS, K)\}, \{(VAL, K)\})$
L	$(\{(NUM, O), (NUM, P)\}, \emptyset, \emptyset)$
M	$(\{(NUM, Q), (NUM, R)\}, \emptyset, \emptyset)$
N	$(\{(NUM, S)\}, \{(PERS, S)\}, \emptyset)$
O	$(\{(NUM, O)\}, \emptyset, \emptyset)$
P	$(\{(NUM, P)\}, \emptyset, \emptyset)$
Q	$(\{(NUM, Q)\}, \emptyset, \emptyset)$
R	$(\{(NUM, R)\}, \emptyset, \emptyset)$
S	$(\{(NUM, S)\}, \{(PERS, S)\}, \emptyset)$

Table 1: Flow analysis - final results

Note that at node B, which corresponds to the first feature structure in the right-hand side of Rule 2, there are five origins for the value of feature NUM. Identifying the rules corresponding to the five nodes, we see, as expected, that they are the lexical entries for determinant, noun and pronoun. For the person, two origins are specified: one corresponds to the left-hand side of Rule 2 (node I) and another one corresponds to the lexical entry for the pronoun *eu* (Rule 11). Considering now node C, we see that there is only one origin for the features NUM and PERS, which is the lexical entry for the verb *dormiu*. In node D, corresponding to Rule 5, no feature value is propagated. By inspecting the grammar, we see that this is due to the fact that all feature values come from the V feature structure on the right-hand side, which is not connected to any other node.

4 Grammar Evaluation and Documentation

Now that the fundamental equations for the flow analysis have been defined, we may use them to define other kinds of analyses that are relevant for grammar evaluation and documentation. One is a *value propagation* analysis, which identifies all the possible values for a feature in a rule and which is mainly concerned with grammar evaluation.

To obtain the set of all possible values for a feature a_k in some feature structure, we simply compute the union of all the values propagated through def-origins associated to it. For example, let us consider once again the grammar of Figure 1. Suppose we want to identify the possible values for feature NUM in the NP constituent of Rule 1. According to the result of flow analysis for node B, given at Table 1, we must find the value of feature NUM in the rules represented by nodes O, P, Q, R, S. Looking at the feature structures for these nodes, we find two possible values: *sing* and *plur*. Repeating the same exercise for feature PERS, we find that the two possible origins give the same value: 3.

An important aspect for the evaluation of the grammar design is the cardinality of the possible values for a feature. If the set of possible values is a singleton, we must consider at least two situations that may explain this result. The lexical entries may not be complete and contain, for example, words that always give the same value for some feature, as is the case in our example for feature PERS in Rule 3. Words that would give another value are simply missing.

In the other situation, there may be different values from the lexicon, but the values are propagated in such a way that some feature in a rule is always instantiated with the same value. In this case, either there is some problem in the grammar design, or the variable used with the feature is irrelevant and should be replaced by the atomic value that is always propagated to it.

Value propagation analysis also informs us about unbound features, that is, features whose set of possible values is empty. For *RHS* nodes, there are two explanations for such a situation. The feature may not exist in any left-hand side of grammar rule. This would happen, for example, if the feature is misspelled. Also, if there is no other feature structure that could be unified with some feature structure on the right-hand side of a rule, then all features in this structure will be unbound, as occurs in our example at node F, where features PERS and NUM will never get a value, since there is no lexicon entry unifiable with this feature structure. In both cases, the result points to some defect in the grammar. For *LHS* nodes, unbound features may result from the unification of features which are unbound for all the *RHS* nodes, again without much of sense. Node D is an ex-

ample of this situation in our example. It could also result from the fact that the intersection of values for all the occurrences of the variable on the right-hand side is empty.

Finally, value propagation analysis may help to identify a rule that would never be used because some variable present in the right-hand side cannot be unified. This happens if the values propagated to two occurrences of the variable are inconsistent. To detect such a situation, we only need to check that the intersection of propagated values is not the empty set.

The second kind of analysis is the *definition-reference* analysis, which provides a way to calculate the different origins for the values of a unifiable feature. For example, it may be useful to identify, for some feature in a specific rule, all the rules and lexicon entries that may be the origin of its value. This analysis is mostly concerned with grammar documentation and impact analysis.

The last kind of analysis, the *reference-definition* analysis, which is also related to documentation and impact analysis, achieves the symmetric purpose: it determines, for a propagated feature value, which other rules this value is propagated to. More specifically, we want to determine whether it is possible that no reference is ever made to the value propagated by feature some feature. Unpropagated definitions in *LHS* nodes could possibly be removed.

5 Experiments

The grammar used in our experimentation is adapted from a Portuguese grammar that has been built for another project (Miola, 2002), where the objective was to test the sensibility of some parsers for phrase structure grammar. The grammar, which contains about 84 rules and uses a lexicon of about 7250 words, recognizes basic sentences. An important characteristic of this grammar is that it has been designed to make it insensitive to common mistakes that could appear in texts written by Brazilians.

Applying our analysis method to this grammar, we constructed a table of results, which is partially illustrated in Figure 2. Each line contains the following information: a rule number, a feature that is contained in the left-hand side of the rule and whose value is a variable and, finally, two sets representing all the possible values and def-origins, respectively. The graph node in the def-origin is identified by a number that corresponds to the rule number in

the grammar. For example, we see that the feature *compl* in rule 100 has two possible values: *sim* and *nao*. Among the origins for the value of this feature, we find rules 230 and 240.

#	Feat	Values	Origins
10	tempo	{fut, imp, pres, ppq, pas}	{(tempo,350), (tempo,360) ... (tempo,1011), (tempo,1016)}
10	numero	{plur, sing}	{(numero,100), (numero,1001) ... (numero,1016)}
10	peessoa	{1, 2, 3}	{(peessoa,110), (peessoa,120) ... (peessoa,1016)}
...			
100	compl	{sim, nao}	{(compl,230), (compl,240) ... (compl,270)}
100	peessoa	{3}	{(peessoa,230), (peessoa,240) ... (peessoa,270)}
...			
180	numero	{}	{}
180	genero	{masc, fem}	{(genero,1005), (genero,1009)}
...			

Table 2: Table of results

With the same data, it is easy to build another table that gives, for every rule and every feature defined in this rule, all other rules to which this value is propagated. Both tables are very useful not only for the identification of defects and idiosyncrasies in the grammar, but also for documentation. Note that in the grammar used in our experiments, a value is always propagated by using the same feature name, but of course another feature name could be used, as allowed in the formalism we presented in section 3.

Table 3 summarizes the cardinality of sets of values resulting from value propagation analysis. It gives the cardinality for every variable occurring in the left-hand side of a rule. We have detected four rules containing a variable that has only one possible value (SNG), and eight rules in which there is a variable whose value is never defined (BOT). In this last case, it means that the right-hand side contains a feature that does not impose any constraint, that is, a useless feature, since its value will never fail with unification.

In Table 4, we consider the different cardinalities for the set of target rules returned by the reference-definition analysis. For each cardinality, we calculated the number of feature values in the grammar whose set of targets has this

Diagnostic	number of variables
BOT	8
SNG	4
MUL	297
Total	309

Table 3: Origins

cardinality. We found 115 occurrences of values that are never propagated to other rules in the grammar, and 23 values that are propagated to only one rule.

Number of target rules	number of features
0	115
1	23
More than 1	73
Total	211

Table 4: Targets

6 Discussion of the results

Our results show that the kinds of analysis we propose produce relevant results that easily identify defects or idiosyncrasies in the grammar used for the experiments.

Incidentally, these results are more informative and precise than those obtained in (Merlo et al., 2004) because of the increased precision produced by the unification sensitive flow model. First, we identified four additional cases of undefined values. For example, consider the rule 180 (see Figure 3), representing noun phrases whose head is a proper name.

The value for feature *numero*(number) is undefined, because it originates from one of the lexical entries for proper names, and is undefined in all entries. In an approach where the union of all values is used, as in (Merlo et al., 2004), this error would not be detected, since both possible values may be propagated to the first feature structure of the right-hand side.

$$\text{NP} \left[\begin{array}{cc} \text{PRO} & \text{nao} \\ \text{GEN} & \text{X} \\ \text{NUM} & \text{Y} \\ \text{DET} & \text{sim} \\ \text{PESS} & 3 \\ \dots & \dots \end{array} \right] \rightarrow \text{DET} \left[\begin{array}{cc} \text{GEN} & \text{X} \\ \text{NUM} & \text{Y} \end{array} \right], \text{PN} \left[\begin{array}{cc} \text{GEN} & \text{X} \\ \text{NUM} & \text{Y} \\ \text{DET} & \text{fac} \end{array} \right]$$

Figure 3: Rule 180 of our grammar

Another interesting result is the distribution of possible values for the feature *pessoa* (per-

son). Table 5 gives the cardinality distribution for this feature. For each cardinality, it gives the number of variable occurrences in the grammar that have this cardinality for its set of possible values.

Cardinality	Number of occurrences
1	1
3	14
4	28

Table 5: Distribution of cardinality for feature *pessoa*

The results show that the distribution of the cardinality for feature *pessoa* is not concentrated in a unique value. There is one singleton, which corresponds to an error in the rule illustrated at Figure 4. The feature *pessoa* in this rule may only have the value 3 (third person). The grammar engineer will quickly realize that this is as expected, since the plural SN represented by this rule is a nominal whose head is a noun, and so cannot take another person. Consequently, we should remove this feature in the right-hand side and fix its value to 3 in the left-hand side.

Let us now turn to the other cardinalities. By

$$\text{NP} \left[\begin{array}{cc} \text{pron} & \text{nao} \\ \text{PRON} & \text{nao} \\ \text{GEN} & \text{X} \\ \text{NUM} & \text{plur} \\ \text{DET} & \text{nao} \\ \text{PESS} & \text{Y} \\ \dots & \dots \end{array} \right] \rightarrow \text{NOM} \left[\begin{array}{cc} \text{GEN} & \text{X} \\ \text{PESS} & \text{Y} \\ \dots & \dots \end{array} \right]$$

Figure 4: Rule 100 of the grammar

closer inspection of the data, we see that in all cases where the cardinality is 3, the set of possible values is $\{1, 2, 3\}$. In all cases where the cardinality is 4, the set has the additional value 0. Since our analysis gives all the origins of the values, we can find easily that when cardinality is 4, the lexical node for pronoun is absent in the set of origins. When the cardinality is 3, the lexical node for pronouns is always present. This leads us to realize that pronouns have three possible values for this feature, whereas verbal forms have four. This discrepancy is certainly worth mentioning, since it could lead to some potential problem in the grammar use.

Let us now turn to the results obtained with the reference-definition analysis, as illustrated in Table 4. Note that since our analysis is driven by the variable occurrences in the grammar, we

cannot at this moment distinguish useless definitions, where a feature value is never unified with another one, from feature values that are unified with at least one instantiated value in the right-hand side of a rule. It is not very difficult to refine our analysis model to distinguish these situations.

After a closer analysis of the data, we discovered 28 useless features among the 115 unpropagated feature values. There are 19 occurrences of values defined in the left-hand side of a grammar for a feature that is never unified with another one. We also found 9 occurrences of features defined in the lexicon that are never propagated in the grammar. In all cases, the information is relevant and points to an error, an incomplete specification in the grammar, or a feature that has been defined in the lexicon for purposes not related to parsing.

Note that the number of singletons is different from the one returned by the definition-reference analysis (23 occurrences instead of 4 occurrences). This is due to the fact that some nodes in the graph represent many lexical entries, and thus may be the origin of different values for the same feature. For example, there is a node representing all the lexical entries for nouns and this node may propagate both values *sing* and *plur* for the feature *numero* (number).

Another type of error that may be identified by the reference-definition is the interruption of the propagation of a value caused by the omission of a feature in some rule. This situation can be detected by searching a feature that is defined in a lexical entry and in a rule. We found two such features in the grammar and in both cases we found errors in some rules.

Finally, the reference-definition analysis is very informative about the importance of some features in the grammar. For example, we discovered that the feature *tempo* (tense) is propagated to 33 rules in the grammar (almost 40% of the rules). This is certainly a valuable information for the grammar designer.

7 Conclusions

Reported results show that the *unification sensitive def-origin* and *value propagation* analyses, which we have presented for evaluating the feature values in a natural language grammar, have identified several value propagation defects in the grammar used for the experiments.

For the same grammar, the *definition-*

reference and *reference-definition* analyses have explicitly recovered the structure of feature propagation and unification, which can be used for documentation purposes and for analyzing the impact of rule modifications and changes. Furthermore, these analyses have also allowed the discovery of errors in the grammar feature propagation and unification structure.

Therefore, the original analyses that have been defined and investigated in this paper seem adequate and useful for grammar evaluation and documentation.

Further research includes the investigation of the performance of the proposed techniques on larger grammars. Also, additional grammar paradigms need to be reflected in the flow analyses. Extensions are also needed to take into consideration aliasing and nested features propagation issues.

References

- A. V. Aho, R. Sethi, and J. D. Ullman. 1986. *Compilers—Principles, Techniques, and Tools*. Addison-Wesley Publishing Co.
- J. Carroll, E. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada.
- Patrick Cousot and Radhia Cousot. 1992. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2-3):103–179.
- S. K. Debray. 1988. Efficient dataflow analysis of logic programs. In ACM Press, editor, *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 260–273, New York.
- Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pages 1420–1425.
- E. Merlo, J.F. Girard, L. Hendren, and R. De Mori. 1995. Multi-valued constant propagation analysis for user interface reengineering. *International Journal of Software Engineering and Knowledge Engineering*, 5(1).
- Ettore Merlo, Michel Gagnon, Dominic Letarte, and Giuliano Antonio. 2004. Feature value propagation analysis for natural language grammars. In *submitted at the Workshop on KNOWLEDGE ORIENTED MAINTENANCE, SEKE 2004 Software engineering and Knowledge Engineering conference*.
- Mariza Miola. 2002. Construção de gramática to português para um estudo comparativo da robustez de alguns algoritmos de análise grammatical. Technical report, Master Dissertation, Universidade Federal do Paraná.
- C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.
- K. Vijay-Shanker and A. Joshi. 1988. Feature structure based tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 714–719.