Grammar Model and Grammar Induction in the System NL PAGE

Vlado Kešelj

Department of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada
`vkeselj@uwaterloo.ca`

Abstract. The input to the natural language parser generation system NL PAGE is in the form of an annotated natural language text, which is used to generate a grammar and a lexicon. This paper describes the format of the text, a new grammar model that is used in the system, and the process of grammar induction.

1 Introduction

Natural language processing contains two separate problems: The first is the linguistic problem of describing a natural language (NL), i.e., of creating an NL grammar and maintaining it. This includes defining an appropriate grammar formalism that can capture phenomena of a significant part of an NL and, still, can keep grammar complexity at a manageable level. The other problem is turning an NL grammar into a parser and providing a way in which the parser can be easily integrated with the rest of a system.

The system NL PAGE (Natural Language Parser Generator) is motivated by a growing need for more efficient and more sophisticated Internet information retrieval. Achieving such sophistication implies consideration of time-and-space efficiency requirements and the ability to produce parsers in various programming languages. The output routines for parsers in C and Java are developed, since C servers and Java applets acting as clients are good candidates for portable and efficient multi-agent systems for Internet information retrieval.

None of the available grammar models and parsers satisfied the requirements stated above, so a new grammar formalism, named FSGM, is defined and the system that induces grammars and generates parsers is implemented.

The FSGM grammar formalism is closely related to the affix grammars over finite lattices (AGFL's, [1]). An AGFL grammar can be easily embedded in the FSGM model. Additionally, it handles movement phenomena, and feature constraints are expressed in a way that provides direct generation of efficient, non-symbolic, low-level code.

On the other side, it is a difficult task for a grammarian to create and maintain a grammar for an NL. This general problem is well-known and a way of solving it is the subject of research in grammar induction. The various approaches to grammar induction can be distinguished depending on the underlying grammar formalisms and the level of supervision during the learning. A frequently used formalism is probabilistic context-free grammar [2], but the grammars in other models are also induced: e.g., constraint grammars [3], statistical decision trees [4], and so on.

A supervised induction method, based on the FSGM formalism, is presented. At the front end of our system, the input is provided in an easily maintainable form—a set of parse trees of sample sentences. The rules can also be explicitly given. NL PAGE extracts grammar rules and the lexicon, which are transformed into a parser. In the process of extraction of grammar rules, some rules are discarded. The process is described in Section 3.

An earlier stage in the system development was presented in [5].

2 Grammar model

The grammar is a context-free grammar (CFG) with binary features. These grammars are traditionally presented as a model with features that can have one of the two values: '+' or '−'. If we specify a set F of all features appearing in a grammar, then instead of '+' and '−' feature values, we can associate each node in a parse tree with one *feature set* that is a subset of F. This subset consists of all features having value '+'.

The binary features are very convenient, since they can be efficiently handled by a computer in the form of a bit array, and they do nicely capture certain intuitive linguistic models, such as AGFL's (Affix Grammars over Finite Lattices [1]). We use only one simple type of *feature constraints:* for example, if we have a rule $X \to Y_1 Y_2 Y_3$ and if the symbols X, Y_1, Y_2, and Y_3 are associated with feature sets F_0, F_1, F_2, and F_3, respectively, then a typical feature constraint is $(F', \{2,3\})$, where $F' \subset F$. It represents the condition $F' \cap F_2 \cap F_3 \neq \emptyset$, which has to be satisfied if the rule is to be applied. There is another type of constraint such as $(F', \{0,2,3\})$ that represents the same condition $F' \cap F_2 \cap F_3 \neq \emptyset$, but also gives a *feature propagation rule:* $F_0 \leftarrow F_0 \cup (F' \cap F_2 \cap F_3)$.

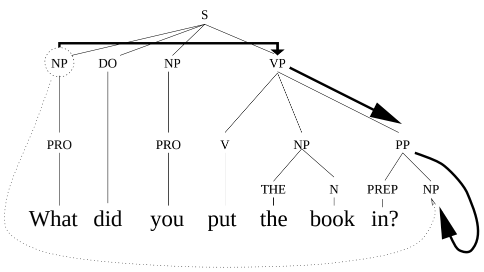

Fig. 1. Movement in our model

Our model also handles movement phenomena. Figure 1 illustrates the way we handle movement of one component: The component is "moved" during parsing along a path in the left-to-right direction, which is called *horizontal movement,* and then in the top-down direction, which is called *vertical movement.* (This is exactly the opposite direction of the movement in generation process.)

Movement in English (and many other languages) is typically handled in a left-to-right direction. Our method is also based on the assumption that the mechanism should handle unbounded movement, while the bounded movement can be handled by additional context-free rules. This solution is also justified by efficiency considerations. This method is novel—it is an intermediate solution between models that do not handle movement explicitly (e.g., GPSG) and more complex mechanisms (e.g., ATN).

Now, that we have described the intuition behind our model, we can formally define our grammar, which we call *Feature-Set Grammar with Movement (FSGM).* The FSGM semantics is explained in the example that follows definition.

Definition 1 (FSGM). *An FSGM is an 11-tuple* $(N,T,P,S,F,\mathcal{F},f_T,f_P,h,v,g)$ *where:*

- (N, T, P, S) *is a CFG having set of nonterminals* N*, set of terminals* T*, set of rules* P*, and a start symbol* $S \in N$*. For any rule* $r = X \to Y_1 \ldots Y_n \in P$*,* $|r|$ $(= n)$ *denotes the number of symbols on the right-hand side, and we call it* rule length. *Having said that* (N, T, P, S) *is a CFG, let us state that we can have two distinct rules in* P *that are identical from the CFG perspective; however, in that case they have to differ in the relations that are defined further.*
- F *is a finite set of features, which are simply some elements.*
- $\mathcal{F} \subset 2^F$ *is the set of distinguished feature sets.*
- $f_T \subset T \times 2^{\mathcal{F}}$ *defines a relation between terminals and sets of feature sets such that:* $(\forall t \in T)(\exists t_f \in 2^{\mathcal{F}})\,(t, t_f) \in f_T$.
- $f_P \subset P \times (\mathcal{F} \times 2^{\{0,1,2,\ldots\}})$ *attaches feature constraints to the rules, so that: if* $(r, (F', I)) \in f_P$*, for a rule* $r \in P$*, then* $I \subset \{0, 1, \ldots, |r|\}$.
- $h \subset P \times \{1, 2, \ldots\}^2$ *defines horizontal movements, so that if* $(r, (i, j)) \in h$*, then* $i, j \in \{1, \ldots, |r|\}$ *and* $i < j$*. For any rule* r *and any three positive integers* i*,* j*, and* k*,* $i \neq k$*, we cannot have both* $(r, (i, j)), (r, (k, j)) \in h$*. This condition makes two horizontal movements with the same destination impossible.*
- $v : P \to \{\perp, 1, 2, \ldots\}$ *defines vertical movements: if* $v(r) = i$*, then* $i =\perp$ *or* $1 \leq i \leq |r|$*. The sign* $\perp$ *is used to denote absence of a movement.*
  For any rule r *and positive integer* j*, if* $v(r) \neq\perp$*, then* $(r, (j, v(r))) \notin h$*. This condition prevents conflicts between horizontal and vertical movements.*
- *Finally,* $g : P \to \{\perp, 1, 2, \ldots\}$ *defines gaps, i.e., the final destinations of moved components. If* $g(r) = i$*, then* $i =\perp$ *or* $1 \leq i \leq |r|$.
  For any rule r *and positive integer* j*, if* $g(r) \neq\perp$*, then* $(r, (j, g(r))) \notin h$*. For each rule* r*, at least one of* $v(r)$ *or* $g(r)$ *has to be equal* $\perp$*. The last two conditions prevent conflicts with horizontal and vertical movements.*

Let us use the example presented in Figure 1 to illustrate the introduced formalism and to explain semantics behind Definition 1: An underlying CFG could be defined as follows:

$$
\begin{aligned}
N &= \{\mathrm{S}, \mathrm{NP}, \mathrm{DO}, \mathrm{VP}, \mathrm{PRO}, \mathrm{V}, \mathrm{PP}, \mathrm{THE}, \mathrm{N}, \mathrm{PREP}\},\\
T &= \{\mathrm{What}, \mathrm{did}, \mathrm{you}, \mathrm{put}, \mathrm{the}, \mathrm{book}, \mathrm{in}\},\\
P &= \{r_1, r_2, \ldots, r_{12}\}
\end{aligned}
$$

where

$$
\begin{array}{ll}
r_1 = \mathrm{S} \to \mathrm{NP\ DO\ NP\ VP} & r_7 = \mathrm{V} \to \mathrm{put}\\
r_2 = \mathrm{NP} \to \mathrm{PRO} & r_8 = \mathrm{NP} \to \mathrm{THE\ N}\\
r_3 = \mathrm{DO} \to \mathrm{did} & r_9 = \mathrm{PP} \to \mathrm{PREP\ NP}\\
r_4 = \mathrm{VP} \to \mathrm{V\ NP\ PP} & r_{10} = \mathrm{THE} \to \mathrm{the}\\
r_5 = \mathrm{PRO} \to \mathrm{What} & r_{11} = \mathrm{N} \to \mathrm{book}\\
r_6 = \mathrm{PRO} \to \mathrm{you} & r_{12} = \mathrm{PREP} \to \mathrm{in},
\end{array}
$$

and

$$S = \mathrm{S}.$$

We want to enforce number and person agreement in the rule r_1. First, we define the feature set

$$F = \{\mathrm{sg}, \mathrm{pl}, \mathrm{p1}, \mathrm{p2}, \mathrm{p3}\},$$

and we distinguish singletons and two additional subsets:

$$\mathcal{F} = \{sg, pl, p1, p2, p3, F_n, F_p\},$$

where $F_n = \{\mathrm{sg}, \mathrm{pl}\}$, and $F_p = \{\mathrm{p1}, \mathrm{p2}, \mathrm{p3}\}$. The singleton sets are denoted by their only elements, e.g., $p1 = \{\mathrm{p1}\}$.

Now, we define f_T, i.e., the terminal feature values:

$$f_T = \{(\mathrm{did}, \{F_n, F_p\}), (\mathrm{you}, \{F_n, p2\})\}.$$

The relation f_T is interpreted as follows: If $(t, t_f) \in f_T$ for a terminal t, then in the parsing process we take the union of all sets in t_f and the obtained feature set is attached to t. If there are more pairs in f_T that are associated with the same terminal t, then the terminal t is ambiguous and we can attach different feature sets to it.

In this example, the terminal 'did' can be associated with the feature set $F_n \cup F_p = F$, and there are no other possibilities.

The following set of constraints provides the necessary bottom-up feature propagation and agreement at the root:

$$
\begin{aligned}
f_P = \{\ & (r_2, (F_n, \{0,1\})), (r_2, (F_p, \{0,1\})), (r_3, (F_n, \{0,1\})), (r_3, (F_p, \{0,1\})),\\
& (r_6, (F_n, \{0,1\})), (r_6, (F_p, \{0,1\})), (r_1, (F_n, \{2,3\})), (r_1, (F_p, \{2,3\}))\}.
\end{aligned}
$$

The interpretation of f_P is: A constraint $(r, (F', I)) \in f_P$ requires that the rule r can be applied only if there is a feature $f \in F'$ that belongs to all feature sets $F_i\,(i \in I)$, where F_0 is the feature set associated with the left-hand-side symbol of the rule r, and F_1, F_2, and so on are feature sets associated with right-hand-side symbols.

In our example: $(r_2, (F_n, \{0,1\}))$ means that the rule NP $\rightarrow$ PRO can be applied if both symbols NP and PRO have at least one feature from F_n in common. In practice, if we build a parse tree in the bottom-up direction, then this constraint is translated to adding all features from $F_n \cap F_1$ to F_0, where F_0 and F_1 are the feature sets associated with the symbols NP and PRO, respectively. We see that the index 0 is treated in a special way because the feature set of the left-hand-side symbol is evaluated according to these constraints and depending on the feature sets of the right-hand-side symbols, i.e., its children. The constraint $(r_1, (F_n, \{2,3\}))$, for example, does not include the index 0, and it represents the constraint $F_n \cap F_2 \cap F_3 \neq \emptyset$.

Finally, let us show how the movement relations have to be defined in order to handle the movement presented in Figure 1:

$$h = \{(r_1, (1,4))\}$$
$$v(r) = \begin{cases} 3 & \text{if } r = r_4 \\ \perp & \text{for all other rules} \end{cases}$$
$$g(r) = \begin{cases} 2 & \text{if } r = r_9 \\ \perp & \text{for all other rules} \end{cases}$$

These relation are interpreted in the following way: If the rule r_1 is applied, then its 1st right-hand-side component (NP) is moved horizontally to the 4th right-hand-side component (VP). When the rule r_4 is applied, a component is moved vertically to its $v(r_4) =$ 3rd right-hand-side component (PP). When the rule r_9 is applied, a component "coming" from the parent node replaces, i.e., fills gap at, the $g(r_9) =$ 2nd right-hand-side component (NP). This can be done only if the moved component has the same category as the gap, and satisfies all constraints that the gap has to satisfy.

Now, we formally define *derivation* in an FSGM (the semantic interpretation is explained after definition):

Definition 2 (Derivation in FSGM). *A derivation in an FSGM is similar to a CF derivation; the only difference is that each terminal and nonterminal is associated with an ordered triple* $(F', m, g) \in 2^F \times (\perp \cup (N \times 2^F)) \times \{\mathrm{gap}, \overline{\mathrm{gap}}\}$*:*

- F' *is the feature set associated with the symbol,*
- m *contains information about a moved component:* $\perp$ *means that no component is moved through this node, otherwise the component category and feature set are given, and*
- gap *denotes that the symbol is a gap filled by a component, otherwise (and more frequently)* $\overline{\mathrm{gap}}$ *denotes that it is not a gap.*

If α *and* β *are two strings of symbols from* $N \cup T$ *associated with such triples, then*

$$\alpha\, X(F_0, m_0, g_0)\, \beta \Rightarrow \alpha\, Y_1(F_1, m_1, g_1) \ldots Y_n(F_n, m_n, g_n)\, \beta,$$

where $(F_0, m_0, g_0), (F_1, m_1, g_1) \ldots (F_n, m_n, g_n)$ *are triples associated with symbols* $X, Y_1 \ldots Y_n$, *is a* derivation *if all of the following hold:*

- $g_0 = \overline{\text{gap}}$ *and there exists a rule* $p = X \rightarrow Y_1 \ldots Y_n \in P$.
- *For every terminal* Y_i *there is a pair* $(Y_i, t_Y) \in f_T$ *such that* $F_i = \cup_{F' \in t_Y} F'$.
- *For all pairs* $(p, (F', I)) \in f_P$, $F' \cap (\cap_{i \in I, i \neq 0} F_i) \neq \emptyset$; *and* F_0 *is the union of sets* $F' \cap (\cap_{i \in I, i \neq 0} F_i)$, *for all pairs* $(p, (F', I)) \in f_P$ *such that* $0 \in I$. *If there are no such pairs, we define* $F_0 = F$.
- *For all* $(p, (i, j)) \in h$, $m_j = (Y_i, F_i)$.
- *If* $v(p) \neq \perp$, *then* $m_0 \neq \perp$ *and* $m_0 = m_{v(p)}$.
- *If* $g(p) \neq \perp$, *then* $m_0 \neq \perp$, $m_0 = (Y_{g(p)}, F_{g(p)})$, *and* $g_{g(p)} = \text{gap}$.
- *Finally, all of* m_i*'s and* g_i*'s*, $0 \leq i \leq n$, *that are not specified by these rules have to have values* $\perp$ *and* $\overline{\text{gap}}$, *respectively.*

Definition 3 (Language generated by an FSGM). *We say that an FSGM generates a language* $L \subset T^*$ *if* L *is equal to the set of all strings* w *such that there exists a sequence of derivations* $S(F', \perp, \overline{\text{gap}}) \Rightarrow^+ w'$, *for some nonempty* $F' \subset F$, *so that after removing symbols marked as* gap *and attached triples from* w', *we obtain* w.

Let us illustrate derivations in FSGM using the example from Figure 1:

$$
\begin{aligned}
&\mathrm{S}(F,\perp,\overline{\mathrm{gap}}) \Rightarrow \\
&\Rightarrow\ \mathrm{NP}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{DO}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{NP}(F_n\cup p2,\perp,\overline{\mathrm{gap}})\,\mathrm{VP}(F,(\mathrm{NP},F),\overline{\mathrm{gap}}) \\
&\Rightarrow^4 \mathrm{PRO}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{did}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{PRO}(F_n\cup p2,\perp,\overline{\mathrm{gap}})\,\mathrm{V}(F,\perp,\overline{\mathrm{gap}}) \\
&\qquad \mathrm{NP}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{PP}(F,(\mathrm{NP},F),\overline{\mathrm{gap}}) \\
&\Rightarrow^5 \mathrm{What}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{did}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{you}(F_n\cup p2,\perp,\overline{\mathrm{gap}})\,\mathrm{put}(F,\perp,\overline{\mathrm{gap}}) \\
&\qquad \mathrm{THE}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{N}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{PREP}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{NP}(F,\perp,\mathrm{gap}) \\
&\Rightarrow^3 \mathrm{What}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{did}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{you}(F_n\cup p2,\perp,\overline{\mathrm{gap}})\,\mathrm{put}(F,\perp,\overline{\mathrm{gap}}) \\
&\qquad \mathrm{the}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{book}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{in}(F,\perp,\overline{\mathrm{gap}})\,\mathrm{NP}(F,\perp,\mathrm{gap}).
\end{aligned}
$$

After removing all triples and the symbol NP, since it is "marked" by 'gap', we get our sentence.

Comparison with AGFL grammars. Any AGFL grammar ([1]) can be directly translated to an FSGM grammar: The set of *terminal affixes* becomes F, *nonterminal affixes* translate to feature sets, and *consistent substitution* translates to our feature constraints. Thus, the AGFL grammar model is in a straightforward way embedded in the FSGM model. The AGFL model provides a convenient way of describing NL's for linguists [1]. Additionally, the FSGM model defines a formalism for movement phenomena and is used to define very efficient parsers.

3 Inducing the grammar from a parse forest

In the previous section we presented our formal grammar model. The presented definition is not very practical from a linguistic point of view. For this reason, an NL-text annotation syntax is developed, which provides a convenient way of modeling an NL. The syntax incorporates the standard parenthesized method of denoting phrase structure of a sentence. It includes some additional constructs for denoting feature constraints and movement. Instead of presenting a tedious definition of the syntax, we illustrate it using the previous example: The grammar presented in the previous example is implicitly defined in the following text:

```
\features: sg,pl,p1,p2,p3
\feature set: Fn={sg,pl}
\feature set: Fp={p1,p2,p3}

(S (NP (PRO What))
   (DO.Fn.Fp did)
   =2Fn =2Fp (NP ^Fn^Fp(PRO.Fn.p2 you))
   =1gap(VP (V put)
            (NP (THE the) (N book))
            ^gap(PP (PREP in) fgap(NP))))
```

Although used syntactically as feature sets, the 'gap' and 'fgap' words actually denote movement relations.

We can define a single CF rule with various feature constraints or movement relations in this way. These are distinct rules in FSGM. It is possible that the same CF rule can be applied under different conditions; this is especially frequent in situations with rules that might or might not include movement. However, this possibility would force us to write all feature constraints whenever we state a rule.

Let us consider the following simplified situation: We define a typical rule `(S (NP...) (VP...))` and annotate it in a number of sentences before concluding that we need a feature constraint, i.e., that the rule has to be formulated as `(S (NP...) =1Fn(VP...))`. Now, we need to rewrite all previously annotated text, since the rule without constraints would always override the one with constraints. In a reverse situation, we may define a rule with all necessary constraints, and further on we want to annotate it (e.g., for documentation purposes) without stating the constraints and be sure that the proper constraints are attached to it.

The previous example illustrates the motive behind the idea that some of the directly generated rules should be discarded. If we have two rules: one without any feature constraints and the same rule with some feature constraints, then the former is said to be *dominated* by the latter rule and so is discarded. Before stating precisely how the rules are discarded, we define the dominance relation in FSGM's:

Definition 4 (Dominance in FSGM's). *Let $(N,T,P,S,F,\mathcal{F},f_T,f_P,h,v,g)$ be an FSGM. We say that a rule $r_1 \in P$ is* dominated *by a rule $r_2 \in P$ if and only if the following conditions are satisfied:*

- *r_1 and r_2 are identical rules in the underlying CFG,*

- *each feature constraint associated with r_1 by f_P is a logical consequence of the constraints associated with r_2,*
- *for all $i, j \in \{1, 2, \ldots\}$, $(r_1, (i, j)) \in h$ if and only if $(r_2, (i, j)) \in h$,*
- *$v(r_1) = v(r_2)$, and*
- *$g(r_1) = g(r_2)$.*

Dominance is obviously a reflexive relation. In the process of grammar induction, we discard all strictly dominated rules (with their associated constraints and movement relations), i.e., we discard all rules r_1 such that there exists a rule r_2 so that r_2 is different from r_1 and r_2 dominates r_1.

The term "logical consequence" in the previous definition is ambiguous since we did not state what assumptions we can use in this logical inference. We use only the constraints associated with rules r_1 and r_2. Hence, nature of the set F, constraints of the other rules, contents of the distinguished feature sets, and so on, are not used.

In this way, the effective evaluation of the dominance relation is easily done, since the only inference rule is:

$$(F_1', I_1) \text{ implies } (F_2', I_2) \text{ if and only if } F_1' = F_2' \text{ and } I_1 \supset I_2.$$

There are no other inference rules, because for any set of feature constraints, such that there are no two constraints related in the above way, it is possible to construct a feature set F and define the distinguished feature sets so that some arbitrarily chosen constraints are satisfied and the rest of them are not.

Due to this, it was easy to implement the dominance relation in our system, which performs well in practice. It is desirable to extend the above definition so that logical inference can be done using all available assumptions about the grammar. However, this problem is NP-complete in general. Let us show, using an example, how the 3-SAT problem can be reduced to our "dominance" problem:

Let an instance of the 3-SAT problem be the constraint satisfaction problem of the following formula:

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \tag{1}$$

First, we define an FSGM grammar:

$$\begin{aligned}
N &= \{\mathrm{S}, \mathrm{A}\},\ \ T = \{\mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3, \mathrm{t}_4\},\ \ S = \mathrm{S},\ \ F = \{\mathrm{x}_1, \ldots, \mathrm{x}_4, \overline{\mathrm{x}}_1, \ldots, \overline{\mathrm{x}}_4\},\\
P &= \{r_1, r_2, r_3\},\ \text{ where } r_1 = \mathrm{S} \to \mathrm{A},\ r_2 = \mathrm{S} \to \mathrm{A},\ r_3 = \mathrm{A} \to \mathrm{t}_1\mathrm{t}_2\mathrm{t}_3\mathrm{t}_4,\\
\mathcal{F} &= \{\text{the set of all singletons}\} \cup \{F_1, F_2, F_3, F_4, G_1, G_2, G_3\},\\
&\quad \text{where } F_1 = \{\mathrm{x}_1, \overline{\mathrm{x}}_1\}, F_2 = \{\mathrm{x}_2, \overline{\mathrm{x}}_2\}, F_3 = \{\mathrm{x}_3, \overline{\mathrm{x}}_3\}, F_4 = \{\mathrm{x}_4, \overline{\mathrm{x}}_4\},\\
&\quad G_1 = \{\mathrm{x}_1, \overline{\mathrm{x}}_1, \overline{\mathrm{x}}_2\},\ G_2 = \{\mathrm{x}_3, \mathrm{x}_2, \mathrm{x}_4\}\ G_3 = \{\overline{\mathrm{x}}_1, \overline{\mathrm{x}}_3, \overline{\mathrm{x}}_4\},\\
f_T &= \{(\mathrm{t}_1, \{x_1\}), (\mathrm{t}_1, \{\overline{x}_1\}), (\mathrm{t}_2, \{x_2\}), (\mathrm{t}_2, \{\overline{x}_2\}), (\mathrm{t}_3, \{x_3\}), (\mathrm{t}_3, \{\overline{x}_3\}),\\
&\quad (\mathrm{t}_4, \{x_4\}), (\mathrm{t}_4, \{\overline{x}_4\})\},\\
f_P &= \{(r_1, (G_1, \{1\})), (r_1, (G_2, \{1\})), (r_1, (G_3, \{1\})), (r_2, (x_1, \{1\})),\\
&\quad (r_2, (\overline{x}_1, \{1\})), (r_3, (F_1, \{0, 1\})), (r_3, (F_2, \{0, 2\})), (r_3, (F_3, \{0, 3\})),\\
&\quad (r_3, (F_4, \{0, 4\}))\},\ \text{and}\\
h&,\ v,\ \text{and } g \text{ translate all rules to } \perp.
\end{aligned}$$

Now, it can be verified that the constraints of the rule r_2 are unsatisfiable. The constraints of the rule r_1 are satisfiable if and only if the formula (1) is satisfiable. Hence, we see that the rule r_1 dominates the rule r_2 if and only if the formula (1) is not satisfiable. Hence, the problem of deciding the dominance relation in the most general case is NP-complete.

4 Parser generation part of NL PAGE

Finally, we briefly mention the generation part of NL PAGE. After a grammar and a lexicon are extracted, the system produces a parser in C or in Java. Those parsers are used in a demonstration multi-agent system for Internet information retrieval [5]. We emphasize that this grammar model is not a purely theoretical model, but it results in a generator that produces efficient NL parsers.

The experimental timing results of two produced parsers in C and Java are presented in Figure 2. The parsers are based on a grammar having 206 rules.

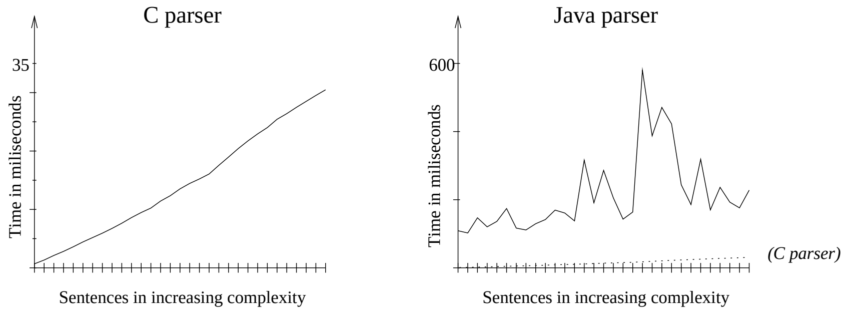

Fig. 2. Timing results for the generated C and Java parsers

Figure 2 gives parsing times of 31 test sentences of increasing complexity. The average parsing time is 14 ms for the C parser and 215 ms for the Java parser, on a Sun SPARCserver 670MP machine.

A training set of 15 sentences was used, and it was included in the set of 31 testing sentences. The sentence length ranges from 3 to 11 words. The generated lexicon contained 100 lexemes and there were no unknown words. All parses were correct.

The experiment is performed on a very small corpus and a very small grammar is generated. Although substantial testing remains, these preliminary results are encouraging.

5 Further directions

We have defined a precise grammar model in Section 2. In Section 3, we have presented an alternative notation, which is used to describe a NL. The notation provides a more flexible way of modeling a NL, leaving to the system to discard some "loose" rules. For future work, we are interested how this boundary between the exact grammar model and descriptive linguistic approach can be pushed towards the latter. The question is: How can a grammar designer to model a grammar by making localized descriptive additions without worrying about remote consequences? In this case, an FSGM would not be directly induced but it would be inferred from the provided data by the system itself. A step further is to provide the system with positive and negative examples of parse trees, and let it infer appropriate feature constraints.

Although we have shown that the general problem of evaluating the dominance relation is NP-complete, it might be useful to make the relation more general. This is a practical issue, which depends on an NL in question.

6 Conclusion

A novel grammar formalism and an associated grammar induction method are presented. The grammar model provides generation of efficient NL parsers, and the induction method provides a tool for creating and maintaining an NL grammar. The experimental results are encouraging, although the testing corpus is very small and larger experiments are part of the future research.

References

1. C. H. A. Koster. Affix grammars for natural languages. In *Attribute Grammars, Applications and Systems, International Summer School SAGA,* Prague, Czechoslovakia, volume 545 of *Lecture Notes in Computer Science.* Springer-Verlag, June 1991.
2. F. Jelinek, J. D. Lafferty, and R. Mercer. Basic methods of probabilistic context-free grammars. In *Speech Recognition and Understanding: Recent Advances, Trends, and Applications. Proceedings of the NATO Advanced Study Institute*, pages 345–360, 1992.
3. Christer Samuelsson, Pasi Tapanainen, and Atro Voutilainene. Inducing constraint grammars. In *Third International Colloquium, ICGI-96*, pages 146–155, September 1996.
4. David M. Magerman. Learning grammatical structure using statistical decision-trees. In *Third International Colloquium, ICGI-96*, pages 1–21, September 1996.
5. Vlado Kešelj. Multi-agent systems for Internet information retrieval using natural language processing. Master's thesis, University of Waterloo, 1997.