Using Active Constraints to Parse GPSGs

Philippe Blache
Institut d'Informatique
Université de Neuchâtel (Suisse)
e-mail : Blache@info.unine.ch

Abstract

Active constraints of the *constraint logic programming* paradigm allow (1) the reduction of the search space of programs and (2) a very concise representation of the problems. These two properties are particularly interesting for parsing problems : they can help us to reduce non-determinism and to use large coverage grammars. In this paper, we describe how to use such constraints for parsing ID/LP grammars and propose an implementation in Prolog III.

Keywords : constraints, syntax, ID/LP formalism, bottom-up filtering, Prolog III

1 Introduction

Logic programming is one of the most useful tools in computational linguistics. These two domains are progressing very rapidly. The former with the emergence of the constraint paradigm and the latter with the systematic use of well-formalized linguistic theories. In the last few years, natural language processing (hereafter NLP) and more precisely syntax have created tools allowing expression of general knowledge.

Constraints simplify parsing problems to a considerable extent, both in a formal and computational way. From a formal point of view, we will see that they allow a very good adequacy between linguistic and computational theories. We know that this property is essential to solve generality, reusability and coverage problems. On the other hand, from a computational point of view, constraints set up a control of the processes which reduces non-determinism in parsing.

The question is to know whether it is possible to implement a parsing method based on actual constraints. The answer depends on the choice of the grammatical formalism. We think that the ID/LP formalism used in GPSG theory can bring a solution to this problem.

In this paper, we will describe a parsing method based on ID/LP formalism using boolean constraints. We will show that this method agrees with the goals of generality and control.

2 Parsing and deduction

Both for historical and formal reasons, parsing has close relations with logic. The birth of Prolog, for example, was conditioned by that and NLP was one of the early applications of this language. One of the reasons, as shown in *[Pereira83]*, is that we can compare parsing and deduction. More precisely, a phrase- structure rule (hereafter PS-rule) can be interpreted as a formula (an implication), like a classical inference rule.

Thus, a PS-rule of the form :

$$SX \rightarrow C_1, \ldots, C_n$$

can be interpreted as the following implication :

$$C_1 \wedge \ldots \wedge C_n \supset SX$$

the clausal form of which is :

$$\neg C_1 \vee \ldots \vee \neg C_n \vee SX$$

Because of the uniqueness of the positive literal, we can interpret a PS-rule as a Horn clause, with a direct translation into Prolog. Thus, a context-free grammar, represented by a set of PS-rule, corresponds to a set of clauses. To verify the grammaticality of a sentence is thus equivalent to proving the consistency of a set of clauses.

There is, however, a restriction in the analogy between PS-rules and clauses : a rule defines an order on its right-hand-side elements, whereas a clause does not. This restriction has important consequences on the generality of the mechanisms. Indeed, the notion of order involves a multiplication of the rules describing a given phrase : we get as many rules as there are configurations. This is one of the limits of phrase-structure grammars.

ID/LP formalism and boolean constraints will allow us to solve this problem. We will obtain a nearly perfect adequacy between the theoretical model and its implementation. Within the classification proposed in *[Evans87]*, it will be a strong direct interpretation of the model.

3 Constraints and linguistic theory

The basic mechanism of constraint logic programming is the restriction of the search space, or the reduction of the domain-variables. This goal can be reached differently depending on the active or passive constraint type (cf *[VanHentenryck89]*). In the classical logic programming framework, the basic technique is that of generate-and-test. In this case, the program generates values for the variables before verifying some of their properties : the search space is reduced *a posteriori*. On the other hand, in the CLP paradigm, the use of constraints allows the reduction of this space *a priori*. Moreover, the set of constraints forms a system which incorporates new constraints during the process, while the use of simple predicates verifying a property only has a local scope.

This active/passive distinction can be useful for parsing, especially according to the type of knowledge that is constrained. Active constraints can easily be defined for syntactic structures and their formation. On the other hand, expressing relations between these structures with this kind of constraint is not always possible.

We will describe the principles governing the formation of the structures. A syntactic structure can be of two types :

- simple structures : lexical categories (e.g. *Det, N, V* ...)
- complex structures : phrases or propositions (e.g. *NP, VP* ...)

The formation of complex structures is governed by two types of knowledge :

- internal : specific information within a structure
- external : relations between structures

Internal knowledge concerns the structure composition, independently of its context. For a phrase, it is the set of its constituents. External knowledge describes interactions between structures. They concern on the one hand the order and on the other hand the government (in the sense of phrase-structure grammars : selection, agreement ...).

ID/LP formalism uses such a distinction : it separates information about immediate dominance (i.e. the set of possible constituents of a phrase) from that on linear precedence (i.e. the partial order relation between these constituents).

It is possible to consider these two types of knowledge as constraints (cf *[Saint-Dizier91]*). But it is important to distinguish their respective functionings. We will illustrate this point by presenting principles for each type.

◇ *Internal knowledge*

Each complex structure must contain at least one particular element called the head. This category gives the phrase its type and its presence is compulsory. The other constituents are usually optional. We must specify that local constraints could require the presence of a particular category, but it is a sub-categorization aspect : it concerns relations between the sub-structures of the complex structure and is not specific to the structure itself. We will see that this distinction between optional and compulsory constituents can be represented directly as an active constraint.

◇ *External knowledge*

In the case of ID/LP formalism, the order constraints (i.e. linear precedence) cannot be easily used with an *a priori* reduction of the search space. Indeed, LP-rules define a partial order upon the set of categories. The LP-acceptability relation uses this order and can be regarded as a constraint upon the domain-variables. It is a symbolic user- defined constraint. The use of this kind of constraint is possible in Chip (cf *[Dincbas88]*), but not in Prolog III (cf *[Colmerauer90]*).

However, using this order relation as an actual constraint allowing the reduction of domain-variables is difficult. In so far as it is a partial order, the LP notion cannot be used to predict the categories that can follow a constituent. It is used during the parse to verify the possibility for each new category to appear at a given place in the syntactic structure.

Generally speaking, internal properties allow an easier use of active constraints than external ones.

4 Constraints and ID/LP formalism

As we have seen, ID-rules of ID/LP formalism only contain the set of possible constituents (without any notion of order). Therefore, an ID-rule is strictly equivalent to a clause.

Example :

$$NP \rightarrow_{id} Det, N, AP \equiv NP \vee \neg Det \vee \neg N \vee \neg AP$$

This equivalence is the basis of the conciseness and generality properties of GPSG. But it is difficult to represent. As we have seen, logic programming cannot directly represent the non-ordered aspect of a clause. However, it is possible to represent this kind of information as active constraints. These must allow the expression of the simple fact that a phrase is well-formed if it is at least composed of the constituents $C_1, \ldots, C_n$. Other relations between the structures (like order or selection) will only be verified if this constraint is satisfied.

Practically, each rule describing a phrase corresponds to a clause whose literals represent categories. An ID-rule is thus translated into a boolean formula where each category corresponds to a boolean. The semantics of this representation is the following :

A literal is true if it corresponds to a well-formed structure. A structure is well- formed if it corresponds to a lexical category (simple structure) or to a well- formed phrase (complex structure).

Thus, the boolean value of a complex structure is the interpretation of this formula, and so depends on the value of its constituents.

Example :

Given the following set of ID-rules describing a NP :

$NP \rightarrow_{id} Det, N$

$NP \rightarrow_{id} N$

$NP \rightarrow_{id} Det, AP, PP, N$

$NP \rightarrow_{id} Det, AP, N$

$NP \rightarrow_{id} Det, PP, N$

This set of rules corresponds to the following formula :

$(Det \wedge N) \vee$

$(N) \vee$

$(Det \wedge AP \wedge PP \wedge N) \vee$

$(Det \wedge AP \wedge N) \vee$

$(Det \wedge PP \wedge N) \supset NP$

It is interesting to note that the ID/LP formalism strongly reduces the problem of PS-rules multiplication inherent in phrase-structure grammars. However, as we have seen in the previous example, there is still a redundancy in the information. Indeed, a set of rules describing a phrase allows us to distinguish between two types of constituents according to their optional or compulsory aspect. Hence, for each phrase we can define a minimal set of compulsory constituents (generally limited to the head of the phrase), which we call the *minimal set* of a phrase.

Example :

In the previous example, the minimal set of the NP is $\{N\}$.

We introduce an additional restriction preventing the repetition of an identical category within a phrase. This restriction is very strong and has to be relaxed for some categories (such as PP). But it remains a general principle : most of the categories should not be repeated.

We then construct a principle defining the well-formedness of complex structures. This principle only concerns internal knowledge :

A phrase is well-formed iff it respects the following properties :

- *it contains at least one head*
- *no constituent is repeated*
- *all its embedded phrases are well-formed*

In the logical paradigm (equivalence between a rule and a clause), we say that a literal is true if it corresponds to a lexical category of the parsed sentence or if it corresponds to a well-formed phrase.

This formation rule allows us to simplify the verification of the grammaticality of a sentence. We simply need to verify the presence of the minimal set of compulsory constituents to indicate the well-formedness of a phrase. The boolean value of the complete structure is then evaluated recursively. If all the intermediate structures are true, the complete structure is also true and corresponds to a grammatical sentence.

We will call *realization* the actual presence of a category in the syntactic structure corresponding to a sentence. The verification process of the well-formedness of a phrase follows these steps

1. verification of the realization of the minimal set
2. verification of the membership of the realized constituents within the minimal set
3. verification of the uniqueness of the constituents in a phrase
4. verification of the well-formedness of embedded phrases

In an active constraint, we replace the set of clauses describing all the possible constructions with a system of constraints S defining the set of possible constituents and the condition of realization for the minimal set. We can represent it as follow :

Let C be the set of possible constituents of a phrase XP, let X be the head of XP, let M be the minimal set such as $M = \{X\} \cup C'$ (where $C' \subset C$), and let Δ be the disjunction of the literals of M. The well-formedness constraint is :

$$S = \{\Delta \supset XP\}$$

Example :

The well-formedness constraint for a NP is : $\{N \supset NP\}$

The well-formedness constraint for a PP is : $\{Prep \wedge NP \supset PP\}$

It is interesting to note that the implication corresponding to the set of rules describing the NP in the previous example forms a system of constraints that can be simplified to $\{N \supset NP\}$. This property is verified for all phrases :

Given a grammar G, $\forall XP$ such that $XP \in G$, let Δ be the disjunction of the literals of the minimal set of XP, then the formula corresponding to the rules describing XP is simplified to $\{\Delta \supset XP\}$.

We thus have both a linguistic and a formal justification of the active constraint used to verify the well-formedness of a phrase.

5 Implementation in Prolog III

We will now describe the parsing strategy and its implementation.

5.1 Bottom-up filtering

Our parsing strategy relies on the concept of left boundary of a phrase. It is an improvement of the left-corner strategy (cf *[Rosenkrantz70]*) called *bottom-up filtering* (cf *[Blache90]*). It consists in using the information extracted from LP constraints to determine all the left-bounds of the phrases from the list of lexical categories corresponding to a sentence. This process, unlike the left-corner one, relies on a distributional analysis of the categories and the verification of some properties.

We define the following functions which allow the initialization of the left boundaries.

◇ *First-legal daughters (noted FLD(P))* : this function defines for each phrase P the set of categories that can appear as left boudaries. It is defined as follows (LP relation between sets is noted with $\prec$) :

Let P be a phrase, $\forall\alpha$ such that $P \to \alpha$ then FLD, the set of first legal daughters, is defined as follows:

$$FLD(P) = \{c \in \alpha \text{ such that } c \prec \alpha - \{c\}\ \}$$

◇ *Immediate precedence (noted $IP_P(c)$)* : this function defines for each FLD c of a phrase P the set of categories that can precede c in P. It is defined as follows :

Let P be a phrase, $\forall\alpha$ such that $P \to \alpha$, let x be a non-terminal, let $c \in FLD(P)$, then $IP_P(c)$, the set of immediate precedence of c for P, is defined as follows:

$$IP_P(c) = \{x \textit{ such that } (x \prec c) \textit{ or } (x \in \alpha \textit{ and neither } x \prec c \textit{ nor } c \prec x \textit{ exist})\}$$

◇ *Initialize* : this function verifies whether a category c is the actual left boundary of a phrase P. It is defined as follow :

Let I be a string, let C be the list of lexical categories of I, $\forall c \in C, c' \in N$ (set of non-terminal symbols) such that c' precedes c in C ;

$$c \textit{ initializes } S \textit{ iff } c \in FLD(S) \textit{ and } c' \notin IP_S(c)$$

The syntactic structure of the sentence is built from a list of partially evaluated structures. The process consists in determining all the left bounds and, from this structure, in completing the partial structures by an analysis of the other constituents of the phrase. This is done by verifying whether the current category can or cannot belong to the current phrase.We have at our disposal the set of possible constituents for each phrase, the LP constraints and the other instanciation principles of the GPSG theory. After these verifications, if the current category cannot belong to the current phrase, then we have reached the right boundary of the current phrase.

Example :

Input sentence :

The old man sings.

Categorization :

Det.Adj.N.V

Partial structure :

S.(NP,Det).(AP,Adj).N.(VP,V)

Complete structure :

(S,(NP,Det,(AP,Adj),N),(VP,V))

This strategy allows a reduction of the search space. Parsing becomes a simple membership test of a category within a set.

5.2 Implementation

The following implementation considers only the ID/LP formalism (instead of the entire GPSG theory). We will not speak here about the other GPSG principles, but their insertion in the ID/LP module is very simple.

The parsing mechanism consists in assigning the value *true* to the booleans corresponding to the categories as and when they appear. If the structure is simple (i.e. a lexical category), the LP-acceptability of this category in the phrase is checked and the corresponding boolean is assigned the value *true*. In the case where the bottom-up filtering detects a left-bound, the corresponding boolean of the current category is assigned the value *true* and the embedded phrase is parsed before coming back to the construction of the current phrase. When we reach the right boundary, the well-formedness of the embedded structures is checked (i.e. all the corresponding booleans must be true). If this is the case, the corresponding boolean value is that of the disjunction Δ of the literals corresponding to the minimal set.

The representation of the categories and their associated booleans will be done through two parallel lists which will be examined simultaneously during an affectation (or any other operation).

A phrase is described by the set of its possible constituents, the set of its optional categories and a formula using its minimal set. The two sets are represented by lists and the formula is an implication of the form $\{\Delta \supset XP\}$. This information is collected into a system of constraints characterizing each phrase.

Here is a simplified version of our parsing process. The following predicates allow the parsing of a phrase and its simple or complex constituents.

It can be noted that the grammatical knowledge is pushed at a low level. It is represented by the set of constraints associated to each phrase. Moreover, at this level we do not use the notion of sub-categorization, but only rules concerning the general structure. We can also notice the conciseness of this representation with regard to classical phrase-structure formalisms.

Description of the implementation

Let G be the following ID/LP grammar :

$NP \rightarrow_{id} Det,\ N$
$NP \rightarrow_{id} N$
$NP \rightarrow_{id} Det,\ AP,\ PP,\ N$
$NP \rightarrow_{id} Det,\ AP,\ N$
$NP \rightarrow_{id} Det,\ PP,\ N$
$NP \rightarrow_{id} Det,\ AP,\ PP,\ N,\ PRel$
$NP \rightarrow_{id} Det,\ AP,\ N,\ PRel$
$NP \rightarrow_{id} Det,\ PP,\ N,\ PRel$
$NP \rightarrow_{id} Det,\ N,\ PRel$
$NP \rightarrow_{id} N,\ PRel$
$VP \rightarrow_{id} V$
$VP \rightarrow_{id} V,\ NP,\ PP$
$VP \rightarrow_{id} V,\ NP$
$VP \rightarrow_{id} V,\ PP$
$AP \rightarrow_{id} Adj$
$PP \rightarrow_{id} Prep,\ NP$
$PRel \rightarrow_{id} Pro,\ NP,\ VP$

The following predicates correspond to the heart of the parser for the grammar G :

```
APhrase(<S(c)>.1,12,Cat,Bool,T) →
      Constituent(S,Cat,Bool)
      LpAcceptable(S,Cat,Bool)
      AnEmbeddedPhrase(<S,c>.1,11,
                        Cat,Bool,A1)
      APhrase(11,12,Cat,Bool,A2)
      Tree(<S[<c>.A1]>.A2,T);
APhrase(<c>.1,11,Cat,Bool,<c>.A) →
      LpAcceptable(c,Cat,Bool)
      Instanciate(c,Cat,Bool)
      APhrase(1,11,Cat,Bool,A);
```

The APhrase rule takes as input the list of partial structures returned by bottom-up filtering. It distinguishes between two cases according to the type of the current structure : complex (rule #1) or simple (rule #2). In the first case, the following processes are called :

- verification of the membership of the current structure within the set of the possible constituents of the current phrase (Constituent rule)
- verification of the LP-acceptability (LpAcceptable rule)
- parse of the embedded complex structure (AnEmbeddedPhrase rule)
- parse of the rest of the phrase (APhrase rule)
- construction and verification of the syntactic tree (Tree rule)

In the case of simple structures, after checking the LP-acceptability, the corresponding boolean is assigned the value *true* (Instanciate rule) and the parse of the current phrase is pursued.

If the APhrase rule fails, the right-bound of the phrase is reached and the parse is pursued at a superior level.

```
AnEmbeddedPhrase(<S,c>.1,11,Cat,Bool,A) →
      Constraints(S,C,B,R,S')
      Instanciate(c,C,B)
      APhrase(1,11,C,B,A)
      CorrectConstituents(R,r)
      Valid(r,S,S',Cat,Bool);
```

The AnEmbeddedPhrase rule allows the parse of a new complex structure. It begins with the system of installing constraints describing this structure (Constraints rule). The validity of the constituents is checked (CorrectConstituents and Valid rules) before returning the boolean value of the parse for this phrase (variable S').

```
Constraints(NP,C,B,R,N_P) →
      { C = <Det,Nm,AP,PP,PRel>,
      B = <D_et,N,A_P,P_P,P_Rel>,
      R = <A_P,P_P,P_Rel>,
      N => N_P };
```

```
Constraints(VP,C,B,R,V_P) →
        { C = <Vb,NP,PP>,
        B = <V,N_P,P_P>,
        R = <N_P,P_P>
        V ⇒ V_P };
Constraints(AP,C,B,R,A_P) →
        { C = <Adj>,
        B = <A_dj>,
        R = <>,
        A_dj ⇒ A_P };
Constraints(PP,C,B,R,P_P) →
        { C = <Prep,NP>,
        B = <P_rep,N_P>,
        R = <N_P>,
        (P_rep & N_P) ⇒ P_P };
Constraints(PRel,C,B,R,P_Rel) →
        { C = <Pro,NP,VP>,
        B = <P_ro,N_P,V_P>,
        R = <N_P,V_P>,
        (P_ro & V_P) ⇒ P_Rel };
```

We can notice that in this representation, subcategorization consists in verifying the boolean values corresponding to the categories concerned.

6 Conclusion

The ID/LP formalism distinguishes between internal and external knowledge about syntactic structures. This characteristic allows the expression of parsing mechanisms at a very high level of generality. We can represent the description of a phrase in an extremely concise way with a rule clustering operation. These properties allow the use of active constraints. The result is an implementation in agreement with the theoretical model respecting in particularl the generality and conciseness properties of GPSG. Moreover, active constraints efficiently control the progress of the processes and limit non-determinism of parsing. This last characteristic is very important for the ID/LP formalism which uses non-ordered rules implying an increase of the search space.

We have shown in this paper how to use active constraints for ID/LP formalism. We can apply the same approach to the entire GPSG theory interpreting features structures and instanciation principles as formulas (cf *[Blache92]*).

The implementation presented here has been done in Prolog III on a Macintosh. From a coverage point of view, we can indicate that the rules in the grammatical formalism presented in our example roughly amounts to twenty standard ID-rules.

References

[Blache90] Blache P. & J.-Y. Morin (1990) Bottom-Up Filtering : a Parsing Strategy for GPSG, *COLING'90.*

[Blache92] Blache P. (1992) Interpretation of GPSG with Constraint Logic Grammars, *ICEBOL'92.*

[Colmerauer90] Colmerauer A. (1990) An Introduction to PrologIII, *CACM*, 33:7

[Damas91] Damas L., Moreira N. & Varile G. (1991) The Formal and Processing Models of CLG, *proceedings of the 5th European Chapter of the ACL.*

[Dincbas88] Dincbas M., VanHentenryck P., Simonis H., Aggoun A. Graf T. & Berthier F. (1988) The Constraint Logic Programming Language CHIP, *International conference on 5th Generation Computer Systems*, ICOT.

[Evans87] Evans R. (1987) *Theoretical and Computational Interpretations of GPSG*, Thesis, University of Sussex.

[Guenthner88] Guenthner F. (1988) Features and Values 1988, *CIS-Bericht-90-2*, München.

[Johnson90] Johnson M. (1990) Features, Frames and Quantifier-free Formulae, in *Logic and Logic Grammars for Language Processing*, P. Saint-Dizier & S. Szpakowicz eds, Ellis Horwood.

[Kasper90] Kasper R. & W. Rounds (1990) The Logic of Unification in Grammar, in *Linguistics and Philosophy*, 13:1.

[Pereira83] Pereira F. & D. Warren (1983) Parsing as Deduction, *ACL83, 21st Annual meeting.*

[Rosenkrantz70] Rosenkrantz D. & P. Lewis (1970) Deterministic Left- corner Parser, *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory.*

[Saint-Dizier91] Saint-Dizier P. (1991) Processing Language with Logical Types and Active Constraints, *proceedings of the 5th European Chapter of the ACL.*

[Stabler90] Stabler E. (1990) Parsing as Logical Constraint Satisfaction, in *Logic and Logic Grammars for Language Processing*, P. Saint-Dizier & S. Szpakowicz eds, Ellis Horwood.

[VanHentenryck89] VanHentenryck P. (1989) *Constraint Satisfaction in Logic Programming*, MIT Press.