GPSG Parsing, Bidirectional Charts, and Connection Graphs

Laurent DEVOS[1,2] Michel GILLOUX[1]

[1] *Centre National d'Études des Télécommunications*
LAA/SLC/AIA
Route de Trégastel, BP 40
22301 Lannion Cedex, France

[2] *École Nationale Supérieure de Sciences Appliquées et de Technologies,*
6, rue de Kerampont, BP 447
22305 Lannion Cedex, France

Abstract: This paper describes a tractable method for parsing GPSG grammars without altering the modularity and expressiveness of this formalism. The proposed method is based on a constraint propagation mechanism which reduces the number of unnecessary structures built at parse time through the early detection of inadmissible local trees. The propagation of constraints is rendered efficient by indexing constraints and categories in a connection graph and by using a bidirectional chart parser together with a bottom-up strategy centered around head constituents.

1. Introduction

Among current syntactic theories, Generalized Phrase Structure Grammars (GPSG) [4] provide an appealing solution for describing natural languages with their modular system of composite categories, rules, constraints and feature propagation principles. As other highly modular systems, GPSG is plagued by the difficulty of designing an efficient algorithm for combining its various components into an executable program. Recent attempts to solve this problem have followed different lines of research.

One solution [9] rests on the almost equivalence between GPSG and Context-Free Grammars (CF). In this framework, GPSG rules, metarules and constraints on well-formed categories and local trees are transformed before any parsing took place into an equivalent set of CF productions for which various efficient parsing methods already exist. In theory, this method has the advantage of preserving the nice expressive properties of GPSG while leading to a parsable grammar. Though, the method is clearly impractical in the case of real grammars due to the enormous number of categories and local trees which must be considered prior to the filtering by the other modules.

A related yet more realistic method [9] follows the same strategy of compiling GPSG into another grammar description language for which there exist parsing algorithms. In this case, the target language [6] has a CF skeleton but is augmented through feature descriptions which provide a natural mapping from GPSG categories. Here the difficulty lies for the linguist in the expression of the grammar in the target language since there exist no algorithm to compile *automatically* the GPSG grammar.

Another set of approaches [1,7,8] attempt to build from scratch a parsing strategy which is suited to GPSG or rather to a modified version of it. The modifications are (sometimes) motivated by linguistic arguments, e.g. enhancing the expressive power, but first to render GPSG computationaly tractable at the expense of some of its distinctive features.

The last class of methods [2,12] try to take GPSG as it is and to design a parsing algorithm which is not only sound but also complete with respect to the theory. Then the difficulty lies in the efficiency of the parser. The method described in this paper belongs to this category. Several principles contribute to its efficiency. First, it is based on a bidirectional chart parser, which, together with the advantages of conventional chart parsing, is well suited to both the ID/LP decomposition and the particular role devoted to head constituants. Second, the various constraints used in GPSG (linear precedence, head feature convention, foot feature principle) are viewed as a set of constraints which are applied as soon as possible, when necessary, to shorten the development of unproductive hypotheses. Third, the constraints are precompiled into a connection graph which minimizes the computation of category subsumption done at run time. And, fourth, the rule invocation strategy is bottom-up and head-driven in that the only ele-

mentary active edges created bottom-up in the chart are those corresponding to head constituants. This strategy allows the FFP and the HFC to be brought into use at the time active edges are created and, consequently, by instantiating very soon a maximum of the foot and head features of dominating categories, to detect violations of constraints long before constituents are completed.

2. Bidirectional chart parsing

Conventional chart parsing techniques [5] avoid redoing identical parses for grammars having a context free basis. In the case of GPSG, these techniques could be straightforwardly customized to accommodate ID rules. This would result in a strategy where an active edge would be created for each category in a rule extending a given inactive edge, be it a head daughter or not.

When one considers the goal of having uninteresting parses fail as soon as possible, it becomes important to make head daughters come into play first. This is because they activate a maximum of constraints on their mother categories by the percolation of features through the HFC and FFP. In order to build local ID trees around their head daughters, the strategy should not force the first constituent attached to an edge to be its leftmost one. We use bidirectional chart parsing [11] because it relaxes this constraint. This technique has been first proposed in the context of speech recognition where island parsing is made advantageous because lexical interpretations are weighed by plausibility factors. Its proponents remarked that it could also be well suited to linguistic theories sharing the notion of head constituent, like GPSG. This intuition was confirmed by the comparison we made on several chart parsing strategies (see section 6 "Implementation").

In order to fit the ID/LP decomposition, each edge of the bidirectional chart must have the following structure :

`Start :`	the leftmost position of the edge;
`End :`	the rightmost position;
`Rule :`	a pointer to the associated ID rule;
`Match :`	a bit vector indicating those members in the LHS of Rule with an extension attached to the edge;
`Category :`	the category of the edge;
`Daughters :`	the list of daughter categories.

Thus an edge is active just in case all bits of Match are set to 1.

New edges are created in two manners. Some are formed when an inactive edge is an extension of a head category in an ID rule (rule 1). They have only one daughter category and we call them *elementary edges*. The others stem from the extension (in the chart sense) of an active edge (rule 2). An active edge A is extended on its right (resp. on its left) any time an inactive edge adjacent on the right (resp. on the left) has a category which is an extension of a category in the Rule of A whose bit in `Match` is not set to 0. To be validated the new edge should not violate any FCR, LP, HFC or FFP constraint (see below).

3. The connection graph

The categories of edges result (1) from the application of feature instantiation principles to an active edge or (2) from the creation of an elementary edge.

Thus the category of a new edge is always an extension of the category of an already existing edge (1) or of the category in the left-hand side (LHS) of an ID rule (2). Since the whole process is initialized by extending lexical inactive edges, the only categories possible for non-lexical edges are extensions of rule LHSs. In addition, LP and FCR constraints only concern categories being an extension of respectively both categories in a LP statement or the left category in a FCR constraint. It is then possible to remark that the constraints that should be verified on a category A are inherited from the constraints to be verified on the category B category A stems from. This is because for A to be an extension of a category C it is necessary that B may be unified with C.

In the same way, inactive edges of category A may only become daughters in local trees whose associated ID rule features a category B in their LHS such that A is an extension of B. Thus determining the ID rules by which elementary edges could be created under the application of rule 1 to an inactive edge may be seen as inheriting these ID rules from the categories of edges leading to it by the recursive application of rule 2 (extension). These two processes, inheriting constraints and ID rules, should be related to the inheritance of connections in a connection graph theorem prover [10].

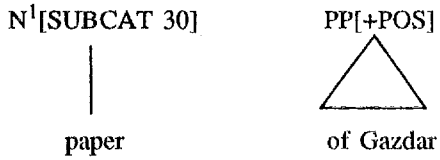
In a connection graph theorem prover (CGTP), resolvents may be only resolved in turn with axioms connected to one of their parents, just like in GPSG categories may be constrained only by constraints concerning their mother category, that is the category they are an extension of. Importing the technique of connection graphs in GPSG allows to reduce the

amount of computation needed to verify constraints. Just like in CGTP, a preprocessing phase, done once for all and independently of any phrase to be parsed, connects every categories in rule right-hand sides (RHS), LPs, FCRs and FSDs to the relevant elements of rule LHSs.

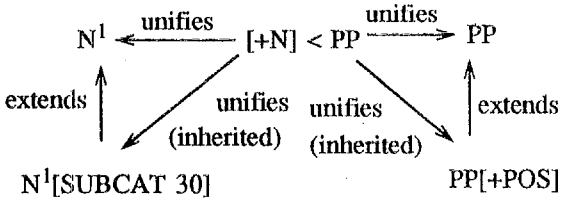
Thus each LP $C_1 < C_2$ (resp. FCR† $C_1 \supset C_2$) is connected to all categories C such as C unifies with C_1 or C unifies with C_2 (resp. such as C unifies with C_1). In the case of FCR, categories C such as C does not unify with C_2 are never created since they violate the FCR.

In the same way, FSDs are connected to categories to help determine where they should apply.

For example, if we consider the ID rule $N^1 \rightarrow$ H, PP and the LP constraint [+N] < PP plus trees



we see that the LP must be checked on this pair of trees because the following connections between the involved categories exist:



On this diagram, two connections have been inherited from former ones.

In order to distinguish the types of constraints connected to categories, several arcs are present in the graph. For example, FCR-Left type arcs connect a category C with a FCR $C_1 \supset C_2$ such as C unifies with C_1.

Inheritance of constraints in the connection graph and creation of bidirectional edges in the chart thus make the bulk of the parsing method. The method is then completely defined through the ways of :

- applying ID rules, constraints and feature instantiation principles to create new edges and categories;
- inheriting connections to ID rules and constraints for each new edge and category;
- computing the initial connection graph.

4. Applying ID rules, constraints, and feature instantiation principles

We assume that, in a first step, all metarules have been applied when possible to base ID rules to generate an extended set of ID rules††. Thus we are left we a set of ID rules, LP and FCR constraints, and feature instantiation principles, included FSDs.

4.1. Immediate Dominance Rules

As remarked in section 3 above, a daughter category in an edge of the chart must be either the category of a lexical item or an extension of an ID rule LHS. Thus, provided that the connection graph was suitably initialized, it is sufficient to consider only those members in ID rule RHSs connected to a lexical item or to an inactive edge in order to create new edges through rules 1 and 2.

4.2. Linear Precedence Constraints

LP statements are checked each time an edge is considered for extension through rule 2. These checks only occur when a new edge A results from B being extended on its left (resp. right) by category C. Then, only those LP, $C_1 < C_2$ such as C is connected to C_2 (resp. C_1) and C_1 (resp. C_2) is connected to some daughter of B are considered. For the selected LPs, the parser has then to compare C_1 and C_2 on one hand and C and daughters of B on the other hand with respect to the partial order relation is-an-extension-of.

4.3. Feature Co-occurrence Restrictions

It is not necessary to wait for an edge to become inactive before checking FCRs. In fact, it is far better to consider them each time a new edge is created. A being the category of the new edge, this requires to check each FCR $C_1 \supset C_2$ connected to the edge in the graph through the extension and unification relations.

4.4. Feature Instantiation Principles

In order to be able to check LP and FCR con-

† We assume all FCRs to be in the canonical form $C_1 \supset C_2$ since the other formats (e.g. $C_1 \equiv C_2$) may be decomposed into a set of canonical forms (e.g. $\{C_1 \supset C_2, C_2 \supset C_1\}$).

†† A better method for applying metarules should trigger them only when needed by the sentence to be parsed. The indexing of constraints in a connection graph could serve this purpose.

straints as soon as possible on new edges, the transmission of features though general instantiation principles (HFC and FFP) is done each time rule 2 is applied. Head and foot features of the extending edge A are compared to those of the extended edge B. There are two cases :

- at least one of these features gets inconsistent values on the daughter A and its potential father B; in this situation, edge A is not valid and the chart is not modified;
- all foot and head features get consistent values; then those which were unspecified or underspecified are set to the suitable value through unification; the category of the new edge is added to the connection graph where it inherits the relevant ID rules, LP, FCR and FSD.

4.5. Feature Specification Defaults

Since feature specification defaults only affect features not set by the feature instantiation principles they only apply to inactive edges. Thus their use consists in applying only those FSD connected in the graph to the category of a new inactive edge.

5. Initializing the Connection Graph

The connection graph is initialized in two phases. A first phase may took place only once for every grammar since it is independent from the particular phrase to be parsed. In this phase, all possible connections between ID rules categories, LPs, FCRs and FSDs are computed. For example, each connection between a rule LHS A and a rule RHS element B such that A unifies with B are memorized.

The second phase depends on the lexical interpretations of the phrase to be parsed and it adds new connections to the graph. These connections are those that hold between a lexical interpretation A and a category B in the RHS of a rule such that A is an extension of B.

6. Implementation

The method has been implemented as a Common Lisp program on a μVAX-II. A core GPSG grammar of French was developed using this parser. In order to assess the interest of the bidirectional chart and its head-driven bottom-up strategy we made a series of tests consisting in parsing a small set of French sentences under three different strategies : strategy I was left-to-right and top-down, strategy II was left-to-right and bottom-up, and strategy III, the chosen one, is bidirectional, bottom-up and head-driven. Table 1 gives the observed performances of the three strategies measured by the total number of edges created and the total CPU time needed to parse all sentences[†].

Strategy	I	II	III
Number of edges	381	189	35
CPU time (in seconds)	238.3	25.1	12.4

Table 1 : A comparison of different strategies

7. Conclusion

The method we propose makes it possible to parse efficiently GPSG grammars but unlike other approaches [1,3,7,8] it allows the grammar to be expressed in the *exact* formalism described by Gazdar and his colleagues in [4] and does not require an intractable preprocessing phase like would do e.g. the compilation of the grammar into an equivalent set of context-free productions.

Although we did not study in detail the suitability of the method when the control agreement principle is taken into account, we do not see any major incompatibility with the use of a connection graph to index the various modules of a GPSG grammar

8. References

[1] Briscoe, T., Grover, C., Boguraev, B., Caroll, J., "A Formalism and Environment for the Development of a Large Grammar of English", 10th International Joint Conf. on Artificial Intelligence, pp. 703-708, 1987.

[2] Busemann, S., Hauenschild, C., "A Constructive View of GPSG or How to Make It Work", Proc. of COLING, pp. 77-82, 1988.

[3] Fisher, A. J., "Practical Parsing of Generalized Phrase Structure Grammars", *Computational Linguistics*, vol. 15, no. 3, pp. 139-148, 1989.

[4] Gazdar, G., Klein, E., Pullum, S., Sag, I., "Generalized Phrase Structure Grammar", Blackwell Publishing, 1985.

[†] CPU time is only an indication of the efficiency of the method since it measures also our implementation. No particular effort was spend on tuning the implementation.

[5] Kay, M., "Algorithm Schemata and Data Structures in Syntactic Processing", in Readings in Natural Language Processing, B. J. Grosz et al. eds, pp. 35-70, 1986.

[6] Kartunen, L., "D-PATR: A Development Environment for Unification Based Grammars", 11th International Conference on Computational Linguistics, p.. 74-80, 1986.

[7] Kilbury, J., "Category Cooccurrence Restrictions and the Elimination of Metarules", Proc. of COLING, p.. 50-55, 1986.

[8] Kilbury, J., "Parsing with Category Cooccurrence Restrictions", Proc. of COLING, pp. 324-327, 1988.

[9] Shieber, S. M., "A Simple Reconstruction of GPSG", Proc. of the 11th International Conference on Computational Linguistics, pp. 145-152, 1985.

[10] Stickel, M. E, "A Nonclausal Connection-Graph Resolution Theorem-Proving Program", Proc. of AAAI-82, 229-233, 1982.

[11] Stock, O., Falcone, R., Insinnamo, P., "Island Parsing and Bidirectional Charts", Proc. of COLING, p.. 636-641, 1988.

[12] Weisweber, W., "Using Constraints in a Constructive Version of GPSG", Proc. of COLING, pp. 738-743, 1988.