A corpus-based technique for grammar development

Philippe Blache, Marie-Laure Guénot & Tristan van Rullen

LPL-CNRS, Université de Provence
29 avenue Robert Schuman
13621 Aix-en-Provence, France

{pb,mlg,tristan}@lpl.univ-aix.fr

Abstract

We argue in this paper in favor of a fully constraint-based approach in the perspective of grammar development. Representing syntactic information by means of constraints makes it possible to include several parsers at the core of the development process. In this approach, any constraint (then any syntactic property) can be evaluated separately. This aspect, on top of being highly useful in a dynamic grammar development, also allows to quantify the impact of a constraint over a grammar. We describe in this paper a general architecture for developing a grammar in this perspective and different tools that can be used in this schema.

1. Introduction

Modern applications, especially in the context of human-machine communication, need to treat unrestricted linguistic material (including spoken languages) with a fine granularity. This means that NLP applications have to be at the same time robust and precise. The first step towards this perspective consists in developing adequate resources, in particular broad-coverage grammars. However, developing grammars remains an important problem and this work is usually done only empirically.

We present in this paper an experiment based on a constraint-based linguistic formalism consisting in developing a broad-coverage grammar by means of corpus-based techniques. More precisely, the approach consists in using different tools, namely a POS-tagger, a shallow parser and a deep parser, for developing an electronic grammar for French taking into account various phenomena and different uses including spoken language syntactic turns. Different parsers are used in this perspective at different stages both in a development and an evaluation perspective. The general idea consists, starting from a core grammar, in parsing previously tagged and disambiguated corpus by means of a deep non-deterministic parser. The results are interpreted in order to identify syntactic phenomena beyond the scope of the grammar. The grammar is then completed and the modifications are experimented over the corpus. The new result is interpreted again in order to examine the consequences of the modifications: new parses, new structures, elimination of unexpected parses, but also false results, spurious ambiguities, etc. This work is then completed with a more systematic evaluation by means of a shallow parser which makes use of the entire grammar, but applies some heuristics in order to control the parse. This parser presents the advantage of being robust and efficient and can be used for parsing large corpora. In our experiment, this parser is used in order to evaluate the different versions of the grammar over different corpora. It is then very easy to compare automatically the efficiency of the grammar, even without any treebank.

2. The formalism of Property Grammars

We think that using a fully constraint-based formalism for representing syntactic information offers several advantages that can have deep consequences in grammar development. More precisely, provided that

information is represented only by means of constraints, it is possible to conceive a grammar development architecture starting from zero and taking advantage of evaluation tools.

In this section, we present the "*Property Grammars*" formalism (hereafter *PG*, see [Blache00]) in which linguistic information corresponds to different kinds of constraints (also called properties). In this approach, parsing comes to constraint satisfaction and, more interestingly in the perspective of grammar development, the result of a parse corresponds to the state of the constraint system. Such a result, called "*characterization*", is available whatever the form of the input, being it grammatical or not. This makes this approach well adapted to the treatment of unrestricted texts and even spoken language corpora. We will give in the following some examples taken from different corpora, from newspapers to dialogs. In the end, we want to show that in such a perspective, different parsing tools can be also used as tools for developing but also evaluating grammars.

In the remaining of this section, we present the *PG* formalism, illustrating the fact that constraints constitute a radically different approach in the perspective of parsing unrestricted texts. In *PG*, different kind of syntactic information corresponds to different kind of constraints. Basically, we use the set of following constraints: *linearity* (linear precedence), *dependency* (semantic relations between units), *obligation* (set of possible heads), *exclusion* (cooccurrence restriction), *requirement* (mandatory cooccurrence) and *uniqueness*:

1. *Obligation* (noted `Oblig`): Specifies the possible heads of a phrase. One of these categories (and only one) has to be realized.
   Example: `Oblig(NP) = {N, Pro}`
2. *Uniqueness* (noted `Uniq`): Set of categories that cannot be repeated in a phrase.
   Example: `Uniq(NP) = {Det, N, AP, PP, Sup, Pro}`
3. *Requirement* (noted ⇒): Cooccurrence between sets of categories.
   Example: `N[com] ⇒ Det`
4. *Exclusion* (noted ≠): Cooccurrence restriction between sets of categories.
   Example: `AP ≠ Sup` (in a NP, a superlative cannot co-occur with an AP)
5. *Linearity* (noted <): Linear precedence constraints.
   Example (in a NP): `Det < N`
6. *Dependency* (noted →): Dependency relations between categories.
   Example (in a NP): `Det → N`

In this approach, all constraints are defined as relations between categories, and all categories are described using a set of properties. Moreover, even if we preserve a hierarchical representation of syntactic information (distinguishing lexical and phrasal levels), the notion of constituency doesn't constitute an explicit information. In other words, categories are only described by a constraint graph in which nodes represent categories involved in the description and edges are relations (or constraints; *cf. fig.1*). The fact that the set of properties describing a category forms a graph is a side effect. It indicates certain cohesion of the description coming from the fact that the description of the category is specified, at the difference of classical generative approaches, not in terms of relations between some constituents and a projection, but only in terms of relations between these constituents. A grammar is then formed by a set of such constraint graphs. The important point, especially in the perspective of grammar engineering, is that all constraints are at that the same level and then independent from each other. This means that it is possible to evaluate them separately. It is also possible to evaluate only a subpart of the constraint system.

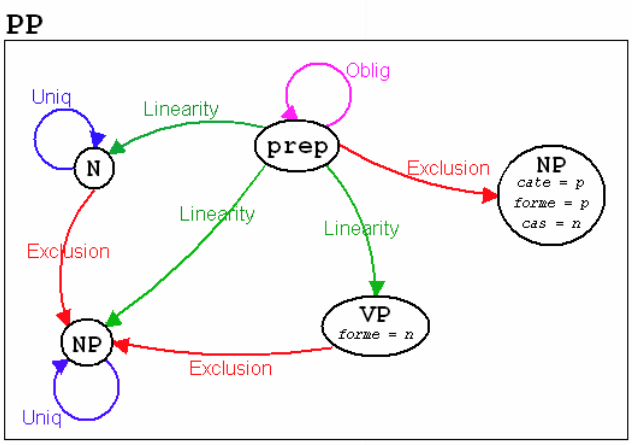


figure 1 : Constraint graph *of the Prepositional Phrase (PP)*
The categories mentioned in the properties of the PP are represented by nodes; each relation (or property) is represented with edges, eventually looping on one node (obligation, uniqueness) or connecting two nodes (requirement, exclusion, dependence, linearity).

Let's describe more precisely the parsing architecture. In *GP*, describing a category consists in finding all the relevant constraints in the grammar and evaluating them. In a data-driven perspective, this comes more precisely to identify a set of categories and verify their properties. At the beginning of the process, the initial set of categories corresponds to the set of lexical categories. The parsing process simply consists in identifying the set of constraints that can be evaluated for the categories belonging to this set. Any constraint belonging to a constraint graph, the evaluation process comes to activate one (or several) constraint graphs. Activating a constraint graph leads to instantiate new categories (which are the categories described by the corresponding graphs).

Activating a graph is then simply the consequence of the evaluation of the constraint system. This means that after evaluation, for each graph, we obtain the set of constraints that are satisfied plus eventually the set of constraints that are violated. These two sets form what we call the *characterization* of a category. We can then describe any kind of input, grammatical (*i.e.* whose set of violated constraints will be empty), or not.

SN_109	La	ligne	aérienne	que	le	docteur	Fergusson	comptait	suivre
nc-fs-	det_1 da-fs--	N_2 nc-fs-	SA_19 afpfs-	Rel_94 pr-----					

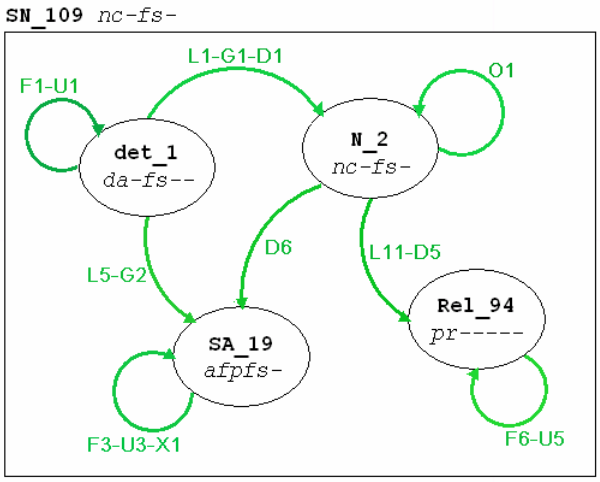


figure 2: Characterization graph
Characterization graph *of the phrase* "The air line that the doctor Fergusson intended to follow". *The nodes correspond to the parsed categories, arrows correspond to the satisfied properties.*

Such an architecture shows in what sense this approach differs from generative ones. In this last case, the general process consists in first building a structure (typically a tree) and then verifying its properties. In our approach, we only use constraint evaluation, without needing any derivation relation nor other extra devices in order to build the syntactic description of an input.

3. The role of constraints in Grammar Engineering

One of the most important aspects of *PG* is that all information is represented by constraints and all constraints are independent. The information comes from constraint interaction, but different kinds of information are represented separately. As explained above, it is then possible to evaluate separately all constraints. We will describe in the next section how it can be possible to take advantage of this aspect in our shallow parsing technique. As for grammar development itself, this characteristic also has important consequences.

Let's first come back on an important point. One of the main differences between a fully constraint-based approach such as *PG* and other classical generative techniques lies in the fact that there is no need to build a structure before being able to verify its properties. More precisely, generative method consists in expressing relations in terms of structures where *PG* uses only relations between objects. This means that a grammar has to be coherent, consistent and complete when using generative formalisms. At least, it should allow to build a structure (a tree) covering an input. Moreover, it is not possible in such approaches to deal easily with partial information. Using *PG* avoids these problems.

First, even when starting from zero the development of the grammar, it is possible to evaluate any subset of properties. This aspect has several consequences on the utility of automatic devices during the development. On one hand, it is possible at any stage of the development of the grammar, to use a parser in order to verify the consequences of new properties, for example in parsing the same corpus and comparing the results of different versions of the grammar. On the other hand, the same devices can also be used in order to evaluate the organization of the grammar itself. All properties being assessable separately, it is possible to evaluate the respective consequences, for example in terms of filtering, for each constraint. This means a possibility of evaluating empirically the power of each constraint as well as specifying this power for a given type of property with respect to the others.

Second, especially when developing a broad coverage grammar, unrestricted texts have to be taken into account. It is then necessary to deal with partial information and partial structures. The constraint-based approach proposed in *PG* relies, as shown before, on graphs which is well adapted to the representation of such information.

Finally, dealing with unrestricted texts leads to treat even ungrammatical inputs. In a classical approach, such data are problematic when using automatic devices: the input can be either rejected (which is not interesting for example when treating spoken languages) or accepted (in this case, a wrong construction is added to the grammar). In our approach, it is possible to introduce a gradient in the grammaticality relying on the possibility of quantifying constraint evaluation. For a given construction, one can give the number and the type of constraints that are violated. Such information is in our opinion as important as the set of satisfied constraints, reason why our *characterization* contains both information. Reciprocally, for a given constraint, it is possible to specify its satisfiability degree in a corpus. And even more generally, it could be possible to specify what kind of constraint is preferably violated. At the difference with the optimality theory (see [Prince93]), we have here a systematic tool for evaluating the respective importance of constraints, generally and locally.

The approach proposed here presents many advantages. It is in particular well adapted for the elaboration of grammars taking into account many different sources (including spoken languages one). Moreover, this approach is highly flexible in the sense that any syntactic information can be evaluated separately by means of tools making it possible to quantify their roles. This approach also presents a very practical interest: it only requires a simple constraint editor (which consists in an interface between an XML file and its representation). It can easily be adapted to the evolution of category representation.

4. Step by step semi automated grammar development

The experimental process of grammar development described in this section is shown hereafter on figure 3 and can be schematised as a versioning step by step semi-automated sequence of experiments.

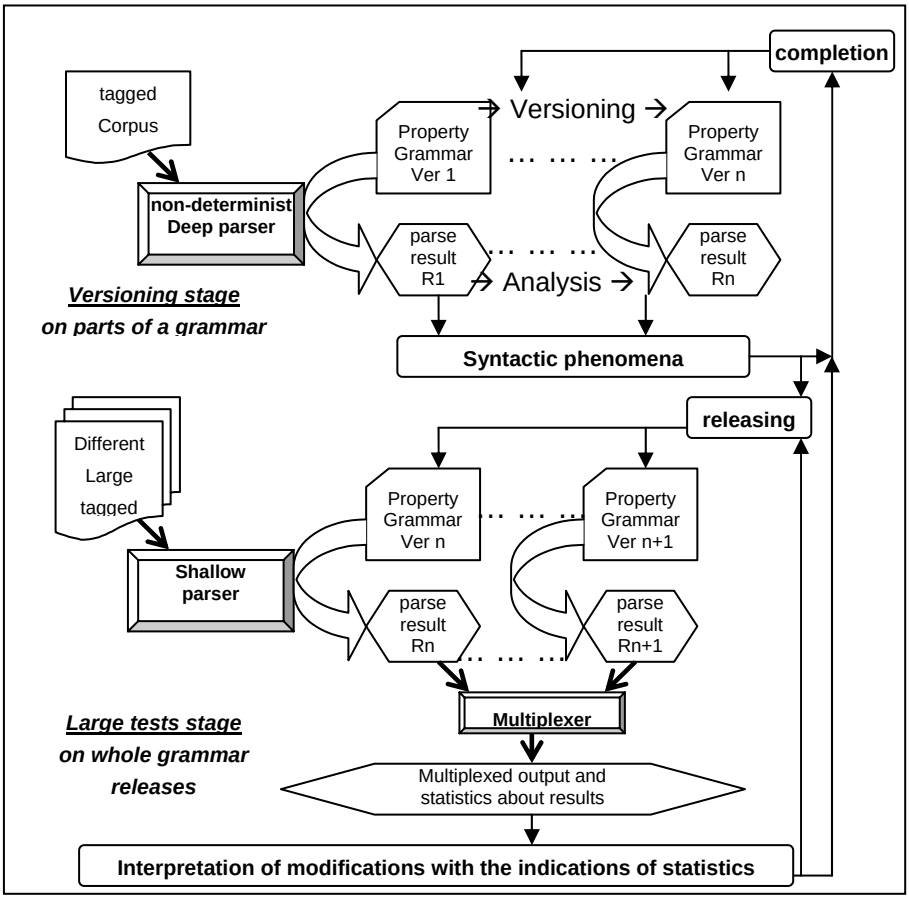


***Figure 3**: step by step semi-automated sequence of experiments*

The development of a broad-coverage grammar for French relies, within our framework, on several points: a basic corpus, a core grammar, and some tools (several parsers). In order to carry out the tests on our grammar, we use a three million words corpus of journalistic texts (hereafter "*corpus-test*"), annotated and disambiguated (the treebank corpus of the LLF, University of Paris 7).

A non-deterministic deep parser can be used as a real grammar development platform. It makes it possible to obtain various results for any input, while entrusting to him either the totality of grammar or one of its subpart, and then being able for example to focus on a subset of properties (to precisely observe their operation, their weight and their incidence on the parse), or on a subset of categories (to focus on the results obtained, the number of non relevant parses, to observe if one can deduce a recurring error pattern and try to avoid it, or on the contrary a recurring satisfactory pattern), without taking into account the rest of the grammar. We can make this type of handling without modifying the structure of neither grammar nor the parser, and that only by neutralizing the non-desired data. The non-determinism of the parser allows to get a mean to evaluate at a first level the efficiency of grammar according to the number and quality of the proposed parses.

The development process can be alternatively carried out in three different ways. First, from a *descriptive* point of view: in this case, we first isolate from the corpus various occurrences of a given construction (*e.g.* clefts, coordination). We then build its fine empirical linguistic description, based on observation of

attested productions, in order to integrate to our basic grammar the properties corresponding to our theoretical conclusions. Then we use a deep parser to analyse our test-corpus and compare the quality of the new results with the ones obtained from the precedent version of the grammar. Such evaluation takes into consideration the evolution of the quantity of parsing proposals according to the modifications of the grammar (*i.e.* conclusions about the consequences of introductions, suppressions and/or corrections, and in which proportions), the evolution of the quality of these proposals (number of relevant parses), the emergence of new suggested structure patterns, etc.

The second development prospect concerns the *coverage* of the grammar. In this case the first stage consists in isolating, among the results obtained with the first version of the grammar, various occurrences of a particular construction (*e.g.* unbounded dependencies). We can then study in details the errors caused by the grammar (omission, imprecision, etc.). Once corrected, we run the parser again with the new version of the grammar to compare the quality of the results.

Thirdly, in order to evaluate the *set of properties* itself and its *semantics*, we can isolate a type of property and observe its behaviour. This use of the deep parser allows to observe directly the impact of a type of property on the results. This can lead to modify the set of constraints itself, *i.e.* their operating mechanism and/or their presence. For example we can compare the quality of one parse result obtained with only one type of property on the one hand, and another one obtained with the other properties on the other hand. Then, while referring to the results obtained with the entire grammar, we can deduce the weight of a type of property on the parsing, either by its presence (for example if the majority of relevant analyses are found in the results obtained with only that one), or by its absence (for example if the majority of relevant analyses are found in the results obtained without this property, and if the number of false results is decreased by its absence). This can lead to an in-depth redefinition of a type of property, for example a modification of its semantics (*i.e.* its satisfaction mechanism), and even its suppression from the grammar (if it appears to be useless, and even more a source of recurrent errors).

Let us consider the parsing results for the following phrase (coming from a dialogue corpus): "*Alors on vous demandera aussi heu pourquoi c'est le meilleur*" ("*so we'll ask you too hum why it's the best*"). One parsing proposition with the first version of the grammar is:

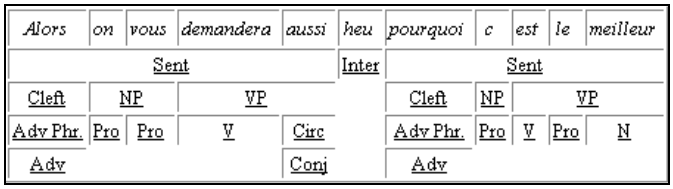


***figure 4**: A parsing proposition with the first version of the grammar*

Following the approach explained above, we have done a lot of changes in the grammar using alternatively one of the three methods described here, and get another parsing proposition with a later version of the grammar:

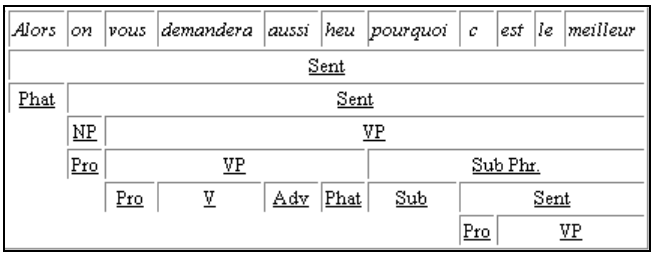


***figure 5**:A parsing proposition with a later version of the grammar*

Each modification of grammar, whatever the method, can have side effects on the rest of the parse (*i.e.* on the elements that we put aside during our different focusings). For example a modification of the constraint graph describing a category, aiming at allowing a particular structure treatment can lead to the proliferation of non-relevant parses for the rest of the corpus (*e.g.* this structure can then be proposed more often, which can cause as a consequence a more significant number of erroneous proposals). This is the reason why we have, for each modification of the grammar, to launch the parse again on the entire test-corpus. If necessary, a new handling of the grammar is to be planned so as to control the effects of the described process. This checking can be done with our deep parser, or with a shallow parser if we want to widen significantly the size of the corpus to parse.

Each new grammar version is subsequently tested over large corpora by means of a shallow parser. The evaluation relies on the analysis of outputs and consists in comparing phrase boundaries of each sentence and in studying local statistics about the width, the nature an the count of parsed phrases. The *Property Grammars* parser used for these tests is deterministic and implements some heuristics in order to control the process. Each sentence is parsed once. The result is built with a stack of categories obtained by means of a classic left-corner parsing, and with a dynamic satisfaction algorithm of the constraint set given by the grammar. The main interest of this parser is that it provides linguistic information about chunks, with hierarchical features, while keeping fastness and robustness as with other shallow parsers.

Thus it is necessary within this grammar development framework to use a tool for comparing parsing results which is fast, complete and efficient. This is the reason why we must handle a test-corpus (so that the provided results can be comparable in a reliable way) and an automatic tool capable of providing quickly and intelligibly some information about common points and differences between results. In this way we can preserve throughout a detailed, step-by-step elaboration, a general point of view on the efficiency of the grammar, which is of primary importance so that the provided results remain homogeneous.

The results of these tests are interesting for grammar development process as for parser improvement. Such a goal is reached with a parameterised automatic evaluation strategy (cf. [Blache02a]) which uses a multiplexer. The technique operates on common boundaries. Its output is the intersection or the difference of the inputs and statistics are given about the relative importance of the categories detected for each border. Set operators such as union, difference and intersection are applied on input borders as well as on categories of compared phrases. This multiplexing method is interesting for an evaluation process without needing a treebank.

The first experiment consists in testing different versions of the grammar over the “treebank corpus”, so that the shallow parser results can be compared for two grammars in order to allow a manual correction of the grammar. The second experiment uses several corpora with different characteristic features such as spoken language, literary style etc. Its aim is to show efficiency of grammars on unrestricted texts. Results improve our knowledge on side effects, limits and the relative importance of properties. Both experiments give statistics about each kind of phrase, their width, their depth, their position. The multiplexed evaluation gives a precise comparison of grammars and information about their main differences. With this tool, we can observe differences and likelihood between two compared grammars in quantitative terms.

The sample results below, obtained on the test corpora, show two grammars letting NPs unchanged, and giving 25% of different VPs and 15% of different PPs.

Grammar	GP1	GP2
Phrases / sentence	19.04	18.97
Words / phrase	1.50	1.50

VP Stats	GP1	GP2
GP1	100%	**75**%
GP2		100%

NP Stats	GP1	GP2
GP1	100%	**100**%
GP2		100%

PP Stats	GP1	GP2
GP1	100%	**85**%
GP2		100%

Figure 6*: Some sample results of a multiplexing process for two grammar releases on a same corpus*

5. Conclusion

This paper describes a technique for empirical corpus-based grammar elaboration. Our approach relies on a fully constraint-based formalism that makes it possible to evaluate precisely each constraint. This is one of the main advantages of such a technique in comparison with classical derivational methods. It allows in particular to evaluate separately any constraint as well as any kind of constraint. This method constitutes then an efficient platform both for developing grammars as well as experimenting and evaluating theoretical results.

The architecture proposed here makes it possible to implement an incremental and step-by-step grammar development environment in which several parsers are used both for developing and evaluating the grammars.

References

[Abeillé03] Abeillé A. (2003), *Une grammaire électronique du français*, CNRS Editions.

[Blache00] Blache P. (2000), "Constraints, Linguistic Theories and Natural Language Processing", in *Natural Language Processing*, D. Christodoulakis (ed), LNAI 1835, Springer-Verlag

[Blache02a] Blache P, Van Rullen T. (2002), "An evaluation of different symbolic shallow parsing techniques", in procs of *LREC-02*.

[Blache02b] Blache P, Guénot M.-L (2002), "Flexible Corpus Annotation with Property Grammars", in procs of the *International Workshop on Treebanks and Linguistic Theories*.

[Butt99] Butt M., T.H. King, M.-E. Nino, & F. Segond (1999), *A Grammar Writer's Cookbook*, CSLI.

[Copestake01] Copestake A. (2001), *Implementing Typed Feature Structure Grammars*, CSLI.

[Kaplan02] Kaplan R., T. King, J. Maxwell III (2002), "Adapting Existing Grammars: The XLE Experience", in proceedings of *Workshop on Grammar Engineering and Evaluation (COLING-02).*

[Kinyon02] Kinyon A. & C. Prolo (2002), "A Classification of Grammar Development Strategies", in proceedings of *Workshop on Grammar Engineering and Evaluation (COLING-02).*

[Prince93] Prince A. & P. Smolensky (1993), "Optimality Theory: Constraint Interaction in Generative Grammars", *Technical Report RUCCS TR-2*, Rutgers Center for Cognitive Science.

[Simov02] Simov K. & al. (2002), "Building a Linguistically Interpreted Corpus of Bulgarian: the BulTreeBank", in proceedings of *LREC-02.*