# Parsing with constraint graphs: a flexible representation for robust parsing

Philippe Blache
LPL-CNRS, Université de Provence
29 Avenue Robert Schuman
13621 Aix-en-Provence, France
pb@lpl.univ-aix.fr

## Abstract

*In this paper, we present an original approach for natural language processing relying on the idea that linguistic information can be represented by means of graphs. One of the interests of this approach is that a linguistic structure does not necessarily cover the entire input: in case of ill-formed sentences, the syntactic structure can for example be constituted by a set of sub-graphs. We propose in this paper a formalism, called Property Grammars, representing the linguistic information by means of constraints represented as graphs. This approach is very flexible and reusable for several applications such as shallow parsing (which consists in using a subset of graphs) or corpus annotation (annotating a text with graph labels). More importantly, it provides a framework for integrating different levels of linguistic information.*

## 1 Introduction

The different domains of linguistic analysis (prosody, syntax, semantics, etc.), even in modern linguistic theories, remain isolated from each others and few approaches try to integrate them within a homogeneous and systematized framework. This leads to a sequential conception of linguistic analysis in which each domain forms a module, the general analysis process being conceived as a succession of treatments.

In most of the cases, linguistic theories are developed for one sub-domain: HPSG in syntax (see [Pollard94], [Sag99]), OT for phonology (see [Prince93]) or DRT for semantics (see [Kamp93]). As a consequence, these theories address the question of the interface with other domains from the point of view of the original domain for which they have been conceived. Such an approach is valid only when a congruence exists between the structures of the different levels (which is generally not the case). In addition, these techniques rely on a complete analysis of all sub-domains, each one playing the role of a stage in the analysis process.

Such a conception is not justified neither from a theoretical point of view (difficulty of specifying homogeneous representations and principles for different domains) nor from a cognitive one. We think preferable to put forth the assumption according to which linguistic analysis (and, in the end, understanding) leans on a dispersed set of partial information coming from different levels (prosodic, lexical, syntactical, pragmatic, etc.).

In this assumption, interaction between domains constitutes the basis of the understanding mechanism. In this case, the process is typically parallel rather than sequential. But our hypothesis goes farther in the sense that each domain can interact with the others and contribute directly to the elaboration of meaning. Such a mechanism leans on partial structures and incomplete information coming from the different domains. Practically, this comes, in a certain sense, to a *fuzzy* conception of parsing.

This notion is used since many years to introduce an idea of degree into categorization (one speaks for example of degrees of nouniness, adjectiveness, etc.) being opposed to a discrete categorization. Several works, following [Zadeh65], have proposed the use of fuzzy context-free grammar. The idea, in the style of probabilistic context-free grammars, consists in associating degrees to rules, which thus can take into account a certain variability. However, such methods require on the one hand an exhaustive

description of the variability and on the other hand do not allow to describe simultaneously different levels of information. It is about an inherent problem in generative approaches, which express the analysis in terms of building structures: for a given utterance, a complete representation (e.g. a syntactic tree) of each domain has to be built. This is not adapted to the general architecture proposed here. In particular this approach doesn't account in a satisfactory way for variability nor non- grammatical constructions (cf. [Blache00a]).

Recent evolutions of constraint-based theories, in particular HPSG (cf. [Sag99]) and OT (cf. [Kager99]), address this problem in similar terms. The idea consists in describing linguistic analysis in terms of constraint interaction. It becomes then possible to build approximate solutions by means of constraints relaxation or under-specification (in this case, the result describes a set of possible solutions). These techniques are well adapted for the processing of non-grammatical utterances.

Unfortunately, implementing linguistic theories using constraint satisfaction techniques is difficult or even impossible. Only few works on this topic can be found (see for example [Duchier99] or [Schröder00]). In the case of HPSG, the problem comes from the fact that constraints are expressed over complex objects (feature structures) requiring a preliminary construction. This is a generative process, which consists in first building a local tree (by means of tree schema) and then verifying its constraints. This preliminary stage is mandatory and relies on tree-like structures, which are not adapted to our needs. Optimality Theory also presents several problems coming in particular from the fact that constraints are generally expressed in terms of relations between an underlying structure and a covert form. Moreover, OT constraints are used to select an optimal form by filtering the set of candidates (generated by a specific function). This approach makes it possible to build approximate solutions for non grammatical structures by means of constraint violation, but constraints are only used in a passive filtering process, not in a truly incremental way (in the sense proposed in [Johnson99]).

We propose an approach meeting the needs of a fuzzy parsing in which (*i*) all linguistic information is represented in terms of constraints and (*ii*) analysis only relies on constraints (without requiring a preliminary process of structure generation).

The formalism we present here, called *Property Grammars*, is situated in this perspective: constraints (also called properties) are stipulated over objects that are directly accessible during the analysis. After a presentation of the formalism, we propose a presentation of Property Grammars leaning on *constraints graphs*. The last section describes an adequate parsing technique.

## 2 Property Grammars

The formalism of *Property Grammars* (noted PG) relies on the idea that any property specified by the description of a language can play the role of a constraint. It is thus a question of collecting and organizing these properties in the form of a constraint system (expressed over categories). Such a system plays the role of a grammar in the sense that it describes the structure of a language. More precisely, it is possible in this approach to indicate the characteristics of an utterance by specifying its properties. Rather than grammaticality, we present this process as an utterance *characterization*.

The representation of syntactic knowledge requires various types of properties, each one corresponding to a specific kind of information. We propose a limited set of relations representing different types of properties: linearity, dependency, obligation, exclusion, exigency, constituency, unicity. This set constitutes the basic syntactic relations of PG. A property is expressed independently from any local structure, but directly between categories. In the following, a predicative representation is given for the constraints, indicating the concerned syntactic unit.

• **Constituency** (*Const*): Defines the maximal set of categories constituting a syntactic unit. This property allows the specification of non lexical categories that appear in the characterization of a given input.

$$(1) \quad Const(NP) = \{Det, AP, N, Sup, PP, Pro\}$$

The example (1) shows the categories (in which *Sup* stands for superlative) that can be constituents of a *NP*.

• **Head** (*Head*): Specifies the set of possible heads (compulsory, unique categories) as in the

example (2) for the *VP*.

(2) $Head(VP) = \{V\}$

In a more formal way, for a given phrase *XP* in a syntactic structure (i.e. the instantiation of an *XP*), let us call *domain of XP*, noted *dom(XP)*, the set of its sub-constituents. The *Head* constraint stipulates that the realization of *XP* must contain a head. It can be described as follows:

$\forall$ *XP*, $\forall$ *dom(XP)*, $\exists x \in Head(XP)$ *dom(XP)* $\cap$ *Head(XP)* = $\{x\}$

• **Unicity** (*Unic*): Set of categories which cannot be repeated in a phrase.

(3) *Unic(NP)* = {*Det, N, AP, PP, Sup, Pro*}

This property is described as follows:

$\forall x \in Unic(XP)$, $x \in dom(XP) \Rightarrow \forall y \in$ *dom(XP)* - $\{x\}$, $x \neq y$

• **Requirement** ($\Rightarrow$): Co-occurrence between sets of categories.

(4) *le, être*$_{[agr]}\} \overset{VP}{\Rightarrow}$ *Clit*$_{[refl,agr]}$

The example (4) describes a property for French which stipulates that when a clitic *le* and a finite verb *être* cooccur (in a *VP*), then a reflexive clitic (which agrees with the verb) is needed (e.g. *Je me le suis dit/ I told that to myself*)

We note in the following description $X \Rightarrow Y$ the requirement property, with X and Y representing sets of categories:

Let $X \overset{XP}{\Rightarrow} Y, X \subset dom(XP) \Rightarrow Y \subset$ *dom(XP)*

• **Exclusion** ($\not\Rightarrow$): Restriction of cooccurrence between sets of categories.

(5) *Clit*[*refl*] $\overset{VP}{\not\Rightarrow}$ *lui*

The example (5) describes a property in French stipulating that in a *VP*, a reflexive clitic cannot cooccur with the clitic *lui* (*Je me lui donne / **I give him myself*).

One can describe an exclusion constraint as follows:

$X \overset{XP}{\not\Rightarrow} Y$ means: if $X \subset$ *dom(XP)*, then $Y \not\subset$ *dom(XP)*

• **Linearity** ($\prec$): Linear precedence constraints.

• **Dependency** ($\rightsquigarrow$): Dependency relations between categories.

These properties contain the basic syntactic information. Other properties can be added if necessary, in particular for integrating knowledge coming from other linguistic domains or for particular devices, for example long distance dependencies.

One can note the use of a dependency property which, as in dependency grammars, concerns a syntactico-semantic relation. It is then possible to express a dependency grammar using the formalism of property grammar. However, several deep differences, among them the use of a hierarchical structure organizing the categories, distinguish these approaches.

In conclusion, a property grammar is formed by a set of (unordered) properties which defines constraints over categories. In an analysis perspective, any subset of categories involved in the description of a given input can be characterized by the constraint system (i.e. the grammar). The problem with the use of a hierarchical structure representation is the accessibility of objects values: the verification of constraint satisfaction requires to know what are the constrained objects (in the case of parsing, the set of categories associated to an input). At the first stage, one know the lexical categories. The phrasal ones can be instantiated thanks to different properties, for example the constituency one which indicates for each category another one that dominates it in the hierarchy. This means that all possible categories forming part of the syntactic structure can be deduced from the set of lexical categories associated to a given input. The characterization of this input is calculated from this maximal set of categories. Building this set of categories doesn't amount to build a local structure, this information is directly accessible from the constituency property for each category.

## 3 Properties as constraint graphs

Each category, whatever its level, is described by a set of properties. These properties are expressed with constraints, therefore in terms of relations between more or less specified categories. It is thus possible to represent the description of a category by a set of relations with other categories. We use for this a graph whose nodes are the categories and the edges the properties connecting them. A grammar is then a set

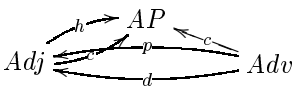
of graphs, each one forming a constraints system associated to a category. We call these graphs describing a syntactic unit *description graphs*: a Property Grammar corresponds to a set of description graphs. More precisely, such graphs being possibly connected to each other (typically by a constituency relation), a grammar is formed by a unique general graph. An analysis consists in finding an assignment (i.e. a set of categories) that can be associated to a sentence to be parsed. The status of this assignment is given by its associated constraints graph.

The schema (6) presents a set of properties describing the adjectival phrase in French:

- $Const(AP) = \{Adj,\ Adv\}$
- $Head(AP) = \{Adj\}$
- $Adj \prec Adv$
- $Adv \rightsquigarrow Adj$

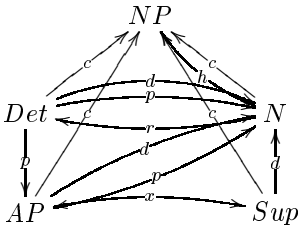
In this graph, each property is represented by a directed edge, the type of the property being indicated by the edge label. In this representation, $c$ stands for constituency, $d$ for dependency, $x$ for exclusion, $p$ stands for precedence, $h$ stands for head, $r$ stands for requirement. For example, the edge $\langle\ Adj \xleftarrow{p} Adv\ \rangle$ represents the linear precedence constraint $Adv \prec Adj$.

(6)




The schema (7) proposes a more complete example of a description graph for the $NP$ in French. One notices that in this approach, all the constraints are at the same level of representation and can be evaluated separately. Furthermore, constrained objects are directly accessible: the categories do not result from a construction operation (in other words, constrained objects are not local structures, such as trees).

(7)



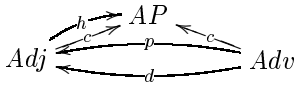

Parsing within the framework of Property Grammars is thus based on a process of constraint satisfaction, constraints being organized in several sub-systems describing syntactic units. In terms of constraint programming, this process consists in finding an assignment for a set of variables, which satisfies a given constraints system. In our case, an assignment is formed by the set of categories associated to the words of an input sentence. For any set of categories, it is possible to determine its description in looking for a sub-graph in which these categories are directly connected. Checking the constraint satisfiability of a set of categories allows to build what we call a *characterization* formed by the graph of relevant constraints with their status (satisfied or not).

The parsing process (more detail will be given in the following section) consists in building the sub-graph of relevant constraints and then evaluating it. This sub-graph describes a syntactic unit which will be considered as a new category to be added to the assignment. We describe in the following the example of the characterization of the input set of categories (8).

(8) $Det \qquad Adj \qquad N$

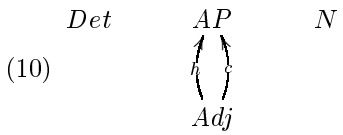
Characterizations evaluated during a parse rely on the description graphs of the grammar. In this example, the first stage consists in building graphs using only lexical nodes. The relevant graph is in this case the sub-graph describing $AP$ with a unique category $Adj$ as presented in (9).

(9)



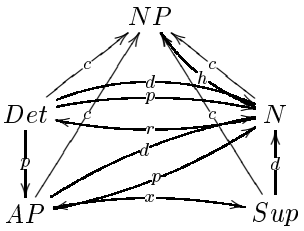

The constituency property plays a particular role: it allows to instantiate the target category of this relation. In this example, the resulting sub-graph extracted from (9) replaces the category $Adj$ of the initial set in the graph (10):

(10)

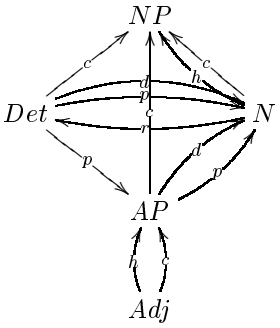



This second stage relies on the characterization of the assignment $\{Det,\ AP,\ N\ \}$. The relevant constraint graph, represented in (11), describes a $NP$ which is determined by the constituency relations between it and these cate-

gories.

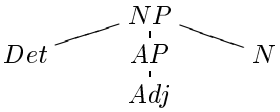
(11)



In this example, as for the preceding stage, all the relevant constraints are satisfied. One obtains finally the characterization graph in (12):

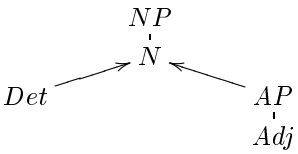
(12)



It is interesting to notice that such a graph explicitly contains all the linguistic information represented in the grammar. It is thus possible to extract the sub-graphs representing each property. In particular, the characterization graph contains two interesting properties: constituency and dependency. The sub-graph of constituency constraints corresponds to the hierarchy relations (i.e. the syntactic tree), and the sub-graph of dependency constraints indicates the dependency tree (in which the relation between a head and its projection is indicated).

(13) -Hierarchy:




-Dependency:




A graph representation allows furthermore to describe simultaneously various levels of linguistic information, which is not possible with a simple hierarchical representation. For example, we know that it is usually not possible to superpose prosodic and syntactic structures (see [Hirst98]). It is thus not possible, except in some cases, to directly associate prosodic information to syntactic constituents. Moreover, as it is the case for syntax, it does not seem relevant to build a unique prosodic structure, represented by a tree of prosodic constituents, covering the entire utterance. It seems to be more interesting to think that prosody is only partially marked in certain places, in the same way as syntax, in certain constructions like dislocated, is not explicit.

Our approach consists in structuring linguistic information by means of atomic objects (the basic nodes of characterization graphs). We use for example as basic nodes a set of positions in the input acoustic signal (e.g. each millisecond). Such an approach is based on the use of *annotation graphs* (cf. [Bird00], [Blache00b]) and consists in proposing a set of edges whose labels have different types (prosodic, syntactic, etc). Our approach relying on graphs, it is possible to specify constituency relations between these positions. A particular node is used, as described in the case of syntactic constituents, as node root which can in its turn take part in the characterization of another constituent. The main interest of using graphs is that it is possible to represent simultaneously any kind of information, without restriction (e.g. projectivity). This is essential for the description of examples such as described in figure () in which syllabic structure is not congruent with the morpho-syntactic one. In this example, the prosodic information, structured into syllables, tonal units and intonational units (see [Hirst98]), is represented by means of relations, in the same way as syntax. It is then possible to specify for a same set of nodes a set of properties.

One can for example represent a set of relations between a syntactic category, a lexical form and a set of phonemes. The interaction between the different analysis levels can be represented by means of properties, such as *Cleft [XP]* $\Rightarrow$ *XP [stress]* indicating that a constituent in a cleft position can be stressed. The specification of the stress position in the *XP* is to its turn described by as set of phonological properties, in the same way as for syntax.

## 4 Implementation

We present in this section a parsing algorithm relying on the use of constraints graphs. The idea consists in looking for all suites of categories

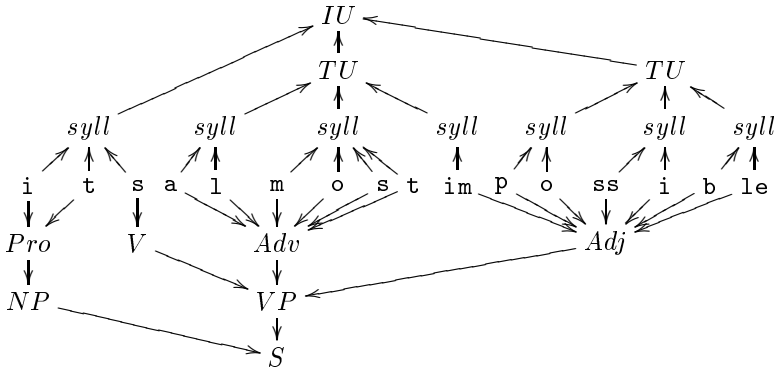



Figure 1: Graph with prosodic and syntactic information

which take part in the description of a syntactic unit. Such categories maintain relations between them and are necessarily connected by a set of edges. In a graph representation, it is only necessary to look for a connected sub-graph included in one of the description graphs of the grammar. This connected graph must have (at least) the set of parsed categories as nodes.

The mechanism consists in building first the sets (that can be reduced to suites) of categories to be parsed. A category is described by a graph in which a particular node is considered as the root and represents the category itself. In case of phrase-level categories, this node is the target of the constituency relations.

The set of categories is thus a set of graphs called *partial graphs*. The process consists in looking for a connected graph in the grammar whose nodes corresponds to the root nodes of the element of the partial graph. When such a graph is found, its root node (more precisely the corresponding graph) is added to the set of categories to be parsed, replacing the set of categories of the partial graph. For example, the analysis of the set $\{Det,\ N\ \}$ consists in looking for a graph containing these categories as nodes. This graph exists, with $NP$ as root node. This category will be instantiated and added to the set of nodes of the sub-graph extracted from the description graph of the $NP$ (cf. schema 7). We call this connected graph a *characterization graph*.

In the following algorithm, the mechanism consists in looking for a characterization graph that covers a subset of the partial graph. In other words, the question is to find a subset of categories that constitutes the entire set of nodes (excluding the root) of an existing characterization graph. In that case, we replace in the partial graph this set of categories (more exactly the set of graphs with these categories as root) with the characterization graph. This process is repeated for all possible partial graphs.

The second stage consists in building the characterization graphs. It thus consists in extracting a connected graph from one of the description graphs and connecting the different elements of the considered partial graph. If such a graph can be built, it is added to the stack of the characterization graph. The algorithm is defined as follows:

- Cat = set of graphs describing categories

- PG = stack of partial graphs

- CG = stack of characterization graphs

- $\wp$(E) = set of subsets of E

- SAT(G) = function extracting the connected graph covering the partial graph G from a description graph of the grammar

- Include(cg, pg) = function indicating whether a sequence of categories belonging to the partial graph pg corresponds to the characterization graph cg.

- Insert(cg, pg) = function building a partial graph formed with a partial graph pg in which the characterization graph cg replaces the set of corresponding categories.

```
1.  Initialization :
2.      PG ← ℘(Cat)
3.      New ← true
4.  While (New) do
5.      New ← false
6.      ∀ pg ∈ PG
7.          ∀ cg ∈ CG
8.              if include(cg, pg) then
9.                  pg2 ← insert(cg, pg)
10.                 push(pg2, PG)
10.                 New ← true
12.     ∀ pg ∈ PG
13.         if SAT(pg) ≠ ∅ then
14.             pop(pg, PG)
15.             push(cg, CG)
16.             New ← true
```

This algorithm has been implemented for the development of a parser for French. Practically, insofar as the parser entirely relies on constraints which constitute the grammar, a result is totally conditioned by the grammar: a broad coverage grammar allows to produce precise analysis. The grammar implemented for French allows the analysis of many different constructions (relatives, comparatives, superlatives, phrasal clauses, coordination, etc.). The parser has been tested on a set of 622 sentences of the newspaper "Le Monde" morphologically annotated by the Talana team (http://talana.linguist.jussieu.fr) and completed with the Cordial POS tagger. A sentence contains an average of 29 words, the system builds about 25 characterization graphs for each, parsing an average of 12 words/second. The result of a parse, encoded in XML, is a set of partially connected subgraphs representing the different characterizations. At this stage, the system is used as a syntactic tagger. An interpreter of the set of characterization graphs is under development.

## 5 Conclusion

An open conception of linguistic analysis, taking into account its different domains and their interactions, offers very rich prospects as well from a formal point of view as from a cognitive one. Our totally constraint-based approach (Property Grammars) constitutes an interpretation framework fully adapted to such a conception. It allows in particular the treatment of incomplete information or analysis of non-grammatical utterances. Constraint satisfaction techniques, together with a constraint graph representation, thus represent a natural and effective paradigm for this approach.

The parser for French developed in this framework is currently tested on a read speech corpus for its syntactic annotation. The result is used to describe constraint interaction between prosody and syntax which will be implemented in the next version of the grammar.

## References


[Bird00] Bird S., D. Day, J. Garofolo, J. Henderson, C. Laprun, & M. Liberman (2000) "ATLAS: A flexible and extensible architecture for linguistic annotation", in proceedings of *LREC*.

[Blache00a] Blache P. (2000) "Constraints, Linguistic Theories and Natural Language Processing", in *Natural Language Processing*, D. Christodoulakis (ed.), Lecture Notes in Artificial Intelligence, Springer.

[Blache00b] Blache P. & D.J. Hirst (2000) "Multilevel Annotation for Spoken-Language Corpora", in proceedings of *ICSLP-2000*.

[Duchier99] Duchier D. & S. Thater (1999) "Parsing with Tree Descriptions: a constraint based approach", in proceedings of *NLULP'99*.

[Hirst98] Hirst D. (1998), "Intonation in British English", in Hirst D. & A. Di Cristo (eds) *Intonation Systems*, Cambridge University Press.

[Johnson99] Johnson D. & S. Lappin. (1999) *Local Constraints vs. Economy*, CSLI.

[Kamp93] Kamp H., U. Reyle (1993) *From Discourse to Logic*, Kluwer.

[Pollard94] Pollard C. & I. Sag (1994) *Head-Driven Phrase Structure Grammars Grammars*, CSLI.

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

[Sag99] Sag I. & T. Wasow (1999), *Syntactic Theory. A Formal Introduction*, CSLI.

[Schröder00] I. Schröder, W. Menzel, K. Foth & M. Schulz (2000), "Modeling Dependency Grammars with Restricted Constraints", in journal *TAL*, 41:1.

[Zadeh65] Zadeh L.(1965) "Fuzzy sets" in *Information and Control*, 8.