# Property Grammars and the Problem of Constraint Satisfaction

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

## 1 Introduction

The notion of constraint appears in all modern linguistic theories. However, constraint satisfaction, which is the typical way of implementing constraints, is not really used for the implementation of such theories. We propose in this paper a formalism called *Property Grammar* which is totally based on constraints: all linguistic information is represented by means of properties which allow parsing to be conceived (and implemented) as a constraint satisfaction problem.

Generally speaking, constraints play a double role : they represent information and form the central part of the resolution process. The first consequence concerns declarativity: representing a problem using constraints is (partly) solving it. Moreover, constraints can be specified at any level of the linguistic description: they can represent local properties (concerning for example lexical selection) or at the opposite universal ones.

These two aspects are of deep importance in the sense that using a constraint-based approach is not only a representational choice, but also concerns parsing. More precisely, there is a complete adequacy between knowledge representation and implementation: when information is represented with constraints, parsing can be seen as a constraint satisfaction process. This idea is clear in some approaches: HPSG (see in particular [Sag99] on the question of constraints and satisfaction), Constraint Dependency Grammars (see [Maruyama90] or [Schröder00]). However, implementation remains difficult in particular because of the accessibility of variables: a constraint satisfaction technique relies on the one hand on the set of variables, and on the other hand on the set of constraints. The result is the state of the constraint system after verifying its coherence. The problem with constraint-based linguistic theories is that the set of variables (i.e. the set of constrained objects) is not immediately accessible. Building a local structure is a prerequisite before verifying constraint satisfiability[1].

*Property grammars* (see [Bès99], [Blache00]) propose a solution to this problem: the use of a local structure in not necessary for the verification of a constraint system. All categories participating to the syntactic description of a given sentence can be determined before parsing. In this case, variables being accessible, the constraint system can be used as such with an enumeration technique. This approach relies on the notion of *characterization* (in place of grammaticality) which allows the verification of properties on all kind of input (including non grammatical ones).

We present in this paper the main characteristics of *Property Grammars* (noted hereafter PG) and we propose a parsing technique relying on constraint satisfaction.

[1] See for example the definition of well formedness given in [Sag99], pp.144-145.

# 2 Property Grammars

In PG, all information is represented by means of properties. The set of properties forms a system of constraints expressed over categories. We present in this section the notions of categories and properties and give some precisions concerning feature value propagation.

## 2.1 Categories

Categories are set of feature-values pairs, as it is usually the case with constraint-based approaches. They represent different kinds of information (e.g. agreement, morphology or semantics) that can be used typically for contextual restrictions. The feature structure (1) presents an example of a verbal entry.

(1)
$$\text{CAT}\ \left[\begin{array}{l} \text{PHON}\ \left\langle \textit{loves} \right\rangle \\ \text{LABEL}\ \text{V} \\ \text{FORM}\ \begin{bmatrix}\textit{finite}\end{bmatrix} \\ \text{AGR}\ \left[\begin{array}{l}\text{NUMBER}\ \textit{sing} \\ \text{PERS}\ \textit{3}\end{array}\right] \\ \text{SEM}\ \left[\begin{array}{l}\text{REL}\ \textit{love} \\ \text{AGT}\ x \\ \text{THEME}\ y\end{array}\right] \end{array}\right]$$

Information encoded in such feature structures specify local properties whereas relations between categories are represented directly with properties (as described in the next section). Constraints applicability is then detemined using subsumption.

In a certain sense, lexical information is minimal in comparison with other theories such as HPSG. Structure sharing for example is taken into account directly by properties. The only propagation between categories occurs between a head and its dominating category. As shown in the next section, other propagation of feature values from one category to another is explicit in the properties.

## 2.2 Properties

We propose a limited set of relations corresponding to different types of properties: linearity, dependency, obligation, exclusion, exigency, constituency, unicity. This set constitutes the basic syntactic relations represented in PG. A property is expressed independently from any local structure, but directly between categories. For clarity, we group properties according to the described syntactic unit, but such a presentation has no operational consequences.

We give in the following a presentation of the properties. A predicative representation is given for these constraints, indicating the concerned syntactic unit. Again, this is not a limitation to a particular local structure, but used only for clarity.

- **Constituency** (*Const*): Defines the maximal set of categories[2] constituting a syntactic unit. This property allows the determination the non lexical categories that will appear in the characterization of a given input.

[2]This set represent the possible types of categories. It is then a set, not a multiset.

*Example*: *Const(NP)* = $\{$*Det, AP, N, Sup, PP, Pro*$\}$

The corresponding categories (in which *Sup* stands for superlative) can be constituants of a *NP*.

- **Obligation** (*Oblig*): Specifies the set of compulsory, unique categories. Such categories correspond to the heads.

  *Example*: *Oblig(VP)* = $\{$*V*$\}$

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

  *Example*: *Unic(NP)* = $\{$*Det, N, AP, PP, Sup, Pro*$\}$

- **Requirement** ($\Rightarrow$): Cooccurrence between sets of categories.

  *Example (for French)*: $\{le, \hat{e}tre_{[agr]}\} \overset{VP}{\Rightarrow} Clit_{[refl,agr]}$
  *Je me le suis dit/ I told that to myself*
  If the clitic *le* and a finite verb *être* cooccur (characterizing a *VP*), then a reflexive clitic (which agrees with the verb) is needed.

- **Exclusion** ($\not\Leftrightarrow$): Restriction of cooccurrence between sets of categories.

  *Example (for French)*: $Clit[refl] \overset{VP}{\not\Leftrightarrow} lui$ (*Je me lui dis / **I told him myself*)
  In a *VP*, a reflexive clitic cannot cooccur with the clitic *lui*.

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

- **Dependency** ($\leadsto$): 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.

A property grammar is formed by a set of (unordered) properties which define constraints over categories. In an analysis perspective, all the subsets 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: we need to know the set of categories associated to an input. The constituency property contains such an information: each category can be associated to another which dominates it in the hierarchy. This means that all the 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 building a local structure: first, this information is directly accessible from the constituency property for each category and secondly hierarchical information between a particular category and an upper-level indicated by such a property is not used in the satisfaction process.

## 2.3 Some precisions

It is important to remark that category structure is independent from the implementation of relations between objects. More precisely, contextual restrictions or selections are encoded in the properties. In the case of subcategorization, for example, a distinction is made between optional and mandatory selection: a preposition requires a noun phrase in a *PP*, an adverb selects a *Det* in a superlative, etc. This information is encoded by means of *requirement* properties. Optional selection is indicated by the fact that concerned categories belong to the same constraint subset (describing a syntactic unit) as the head (constituency, dependency). However, a lexical item can specify an imperative valency. In such cases (for example a transitive verb for which an intransitive construction is impossible), a reference to a specific subcategorization scheme (as in GPSG) can be used.

As shown in the previous section, feature values are used to specify properties. Syntactic information in property grammars being hierarchized, these values has to be accessible at any level. Features are then propagated from the head to the dominating category. This allows the representation of relations between all kind of categories. For example, the representation of the subject/verb relation can be expressed in a property such as:

(2) $$SN \begin{bmatrix} \text{CAS } \textit{nom} \\ \text{SEM } \boxed{1} \end{bmatrix} \leadsto SV \begin{bmatrix} \text{SEM } \begin{bmatrix} \text{AGT } \boxed{1} \end{bmatrix} \end{bmatrix}$$

In the same way, control phenomena can also be represented by means of properties. The example (3) presents a dependency constraint indicating subject sharing between finite and infinitive verbs.

(3) $$V \begin{bmatrix} \text{VFORM } \textit{inf} \\ \text{SEM } \begin{bmatrix} \text{AGT } \boxed{1} \end{bmatrix} \end{bmatrix} \leadsto V \begin{bmatrix} \text{VFORM } \textit{fin} \\ \text{SEM } \begin{bmatrix} \text{AGT } \boxed{1} \end{bmatrix} \end{bmatrix}$$

Such examples illustrate the fact that information is represented by properties, the lexical information being static. Cases of lexical selection can also be represented with a general dependency (or in some cases a requirement) property stipulating that a particular feature must be shared between a head and its complement.

## 2.4 Characterization

In PG, the notion of grammaticality is not crucial. More precisely, grammaticality is replaced with the general notion of *characterization* formed by the set of properties that can characterize an input. Concretely, the characterization of an input corresponds to the state of the constraint system at the end of the parse. Such a result can be interpreted as a unique structure (for example a tree), but not necessarily: some inputs can be characterized with several disconnected structures. The main consequence is that parsing is not restricted to grammatical inputs.

Satisfiability consists then of the following mechanism: knowing the set of categories that can be associated with an input, a characterization is the state of the constraint system formed by the relevant constraints (i.e. the constraints that can be evaluated) for this set of categories. The interest of such a conception is that no notion of well-formedness is used. It can be the case that all constraints are satisfied, but this is not an imperative condition. All kind of input can consequently receive a characterization.

## 3 Examples

We present in this section two simple examples illustrating the role of properties. These examples propose a toy description of the *NP*, in english and in french. All properties are indexed and grouped (according to the categories they describe) for readability.

The example described in figure(1) presents the main properties for the *NP*, including superlative. The *Const* property indicates categories that can appear as constituent of a *NP*. This grammar integrates the construction using a *PP[of]* as a possible head (as in "*the most interesting of the books*") by adding this category to the *Oblig* properties. In this case, as in the case where the *AP* is the head (e.g. "*the small*"), exclusion properties have to be stipulated (constraints 8 and 9). In the same way, the superlative description is distinguished from other *AP* by the requirement property (3) which stipulates that a degree adverb is mandatory (without being a possible head).

Properties of the *NP*

(1) $Const = \{Det, N, AP, Sup, PP[\,of]\}$
(2) $Oblig = \{N, AP, PP[of]\}$
(3) $N[com] \Rightarrow Det$
(4) $Det \prec N$ (5) $AP \prec N$ (6) $Det \prec AP$
(7) $Det \prec Sup$ (8) $N \prec Sup$
(9) $AP \not\Leftrightarrow Sup$
(10) $N \not\Leftrightarrow PP[of]$
(11) $Det \leadsto N$, (12) $AP \leadsto N$, (13) $Sup \leadsto N$

Properties of the *Sup*

(1) $Const = \{Adv[deg], Adj\}$
(2) $Oblig = \{Adj\}$
(3) $Adj \Rightarrow Adv[deg]$
(4) $Adv[deg] \prec Adj$
(5) $Adv[deg] \leadsto Adj$

Figure 1: A property grammar for the English *NP*

The grammar of the French *NP*, also integrating the superlative, presents the interesting particularity of being more ambiguous: same subsets of categories can appear as constituents of different phrase-level categories. In particular, the difference between *AP* and *Sup* is the imperative presence of the *Det* stipulated in constraint (3). This ambiguity will be illustrated in the next section.

Properties of the *NP*

(1) $Const = \{Det, N, AP, Sup\}$
(2) $Oblig = \{N, AP\}$
(3) $N[com] \Rightarrow Det$
(4) $Det \prec N$ (5) $Det \prec AP$
(6) $Det \prec Sup$ (7) $N \prec Sup$
(8) $AP \not\Leftrightarrow Sup$
(9) $Det \leadsto N$ (10) $AP \leadsto N$ (11) $Sup \leadsto N$

Properties of the *AP*

(1) $Const = \{Adj, Adv\}$
(2) $Oblig = \{Adj\}$
(3) $Adv \prec Adj$
(4) $Adv \leadsto Adj$

Properties of the *Sup*

(1) $Const = \{Det, Adv, Adj\}$
(2) $Oblig = \{Adj\}$
(3) $Adj \Rightarrow Det$
(4) $Adj \Rightarrow Adv$
(5) $Det \prec Adv$ (6) $Det \prec Adv$ (7) $Adv \prec Adj$
(8) $Det \leadsto Adj$ (9) $Adv \leadsto Adj$

Figure 2: A property grammar for the French *NP*

# 4 Parsing Technique

We describe in this section a general mechanism implementing the PG characterization process. A grammar being a set of constraints, parsing an input requires first the specification of the set of relevant categories (i.e. for this particular input) constrained by properties. Characterizing an input can be implemented in several ways, using more or less actively constraints. More precisely, constraints can be considered as imperative or at the opposite systematically relaxed. In the first case, characterization is restricted to grammaticality whereas in the last one, all kinds of input can be characterized. For both cases, the core process remains the same. We detail in this section different stages in the parsing mechanism, without specifying the degree of constraint relaxation.

## 4.1 Building the Set of Variables

The first stage consists in determining all the categories that can be associated to each word of the input. This operation can be controlled with disambiguation techniques. However, in the general case, all possible categories can be associated to the input, without any consequence (other than complexity) on the general mechanism. Each category is then associated to a position.

The second step relies on constituency properties and consists in associating each constituent to its dominating category (called *dominant*) in the hierarchy. This is not a classical projection operation between a head and its mother in the sense that all categories are associated with their dominants: a *Det* introduces a *NP*, in the same way as a *N* does. In the end, all input words are associated with a set of categories.

Some restrictions or controls can be proposed at this level, avoiding cycles introduced by recursivity. For example, one can choose to project a given amount of categorie. Another solution is to repeat the projection operation until reaching categories immediately dominated by *P*.

In the following examples, relying on the grammar (1) described in the previous section, the set $\Sigma'_1$ contains all the possible categories that can be used in characterizing the *NP* in (1-a).

(1) a. *the most interesting book*

b. $\Sigma_1 = \{ Det_1, Adv_2, Adj_3, N_4 \}$

c. $\Sigma'_1 = \{ Det_1, NP_1, Adv_2, Sup_2, Adj_3, Sup_3, N_4, \mathrm{NP}_4 \}$

The same kind of example can be given for a similar construction in French (following the grammar given in the previous section), with a higher degree of ambiguity.

(2) a. *le livre le plus vieux. (the oldest book)*

b. $\Sigma_2 = \{ Det_1, Pro_1, N_2, V_2, Det_3, Pro_3, Adv_4, Adj_5 \}$

c. $\Sigma'_2 = \{ Det_1, Pro_1, NP_1, Sup_1, N_2, NP_2, V_2, Det_3, Pro_3, NP_3, Sup_3, Adv_4, AP_4, Sup_4, Adj_5, AP_5, Sup_5, \}$

In both examples, the set of all the possible categories is directly built, without needing the construction of any local structure.

## 4.2 Building Suites

The second stage consists in determining all suites that can be characterized. The most general case, without any control, consists in building all the subsets from the set of possible categories described in the previous section. This means building arrangements of any length (up to the number of positions) according to the possible categories for each position.

Following previous examples, the suite sets are :

(3) a. $S_1 = \{\{\, Det_1,\, Adv_2\},\, \{Det_1,\, Sup_2\},\, \{\, Det_1,\, Adv_2,\, Adj_3\}...\}$

b. $S_2 = \{\{\, Det_1,\, N_2\},\, \{Det_1,\, NP_2\},\, \{\, Det_1,\, N_2,\, Sup_3\}...\}$

These suites cannot be considered as local structures in the sense that the building operation is an enumeration, without any linguistic control. In the most general case (i.e. without any control) one can consider that building suites consists in generating objects before verifying their properties. Such a process can be seen as a filtering one. One of its interests is that it can be used as a pre-parsing or chunking technique, as described in [Blache99]. But one can consider useful (for actual parsing applications) to control such a process: it is then possible to apply constraint verification also during this operation in order to reduce as much as possible the set of possible suites.

## 4.3 Characterizing Suites

A characterization, as explained earlier, is the state of relevant constraints after verification for a given suite. The general process consists then in verifying the properties of each suite. Relevant properties are that which categories constitute a subset of the suite to be characterized. This point illustrates the fact that no derivation is used during the parsing process, and grammar actually plays the role of a constraint system in the sense that constraints (properties) are not dependent from the category they describe. In other words, the hierarchization of the syntactic structure do not rely on the necessity of building local structures (i.e. applying a derivation), but simply indicates the possibility of stipulating relations at different levels.

The following examples present some suite characterizations. A characterization is represented as the sets of satisfied ($P^+$) and unsatisfied constraints ($P^-$). These sets are formed by the index of the constraints in the grammar.

(4) $\mathrm{Det_1/N_2}$ : $P^+(NP) = \{1, 2, 3, 4, 10, 11\}$ $P^-(NP) = \emptyset$

In the example (4), characterizing a suite from $S_1$, all relevant constraints are satisfied, in particular obligation (2), linear precedence (4), requirement (3), exclusion (10).

(5) $\mathrm{Det_1/Sup_2}$ : $P^+(NP) = \{1, 6, 10\}$ $P^-(NP) = \{2\}$

Example (5) presents a suite not positively characterized. In this case, the obligation property (stipulating the presence of a head in a suite) is not satisfied.

(6) $\mathrm{Adv_4/Adj_5}$ : $P^+(AP) = \{1, 2, 3, 4\}$ $P^-(AP) = \emptyset$
$P^+(Sup) = \{1, 2, 4, 7, 9\}$ $P^-(Sup) = \{3\}$

```
Begin

    1. Initialization :
            Active_Edges ← Characterizations
            Inactive_Edges ← ∅
    2. ∀ AE ∈ Active_Edges {
        2.1. ∀ Cat ∈ AE
            2.1.1. if (Non_lexical(Cat)) {
                2.1.1.1. Edge ← suite of juxtaposed Cats in AE
                2.1.1.2. if (Edge ∈ Inactive_Edges) Reduce Edge to Cat
            }
        2.2. if (all categories reduced) Inactive_Edges ← AE
        2.3. ∀ IE ∈ Inactive_Edges {
            2.3.1. Edge ← append(AE, IE)
            2.3.1. Active_Edges ← Characterization(Edge)
        }
    }

End
```

Figure 3: The interpretation algorithm schemata

Example (6) shows an ambiguity (this example is taken from the set $S_2$, relying on the grammar (2). More precisely, a same suite can be characterized on two different ways: this suite satisfy all relevant properties for the *AP*, and violate the requirement property (3) for the *NP*.

As for building suites, such a process can be conceived in the general case without any restriction (i.e. relaxing all non satisfied constraints) or at the opposite controlled by constraint satisfaction. It is even possible to tune the level of relaxation in using a threshold indicating the level of accepted non-satisfied constraints. When this threshold is 0 (i.e. no constraint violation is allowed), then only grammatical suites are characterized.

### 4.4 Interpreting Characterization

The set of characterized suites can be used in many different ways: they can describe some particular sequences from the input; they can also constitute the input of a parsing process. In this case, one solution consists in using this high-level information in a chart-based process. The idea consists in considering all characterized suites as active edges and to apply an iterative process building incrementally inactive edges. The goal of this process is to find a set of juxtaposed suites that can cover the input. In the case where no grammatical coverage can be found, the set of characterized suites allow to propose a set of suites violating some constraints. Again, it is possible to use a threshold (i.e. the amount of violated constraints). In the following, we will present the general mechanism, which does not take into account this possibility.

Figure (3) presents the schemata of the algorithm interpreting the set of characterizations. The first step consists in initializing the set of active edges with the set of characterized suites (an edge is a characterized suite). The restriction to grammatical suites can be applied at this level (and in the step 2.3.1).

For each active edge, the process consists in verifying whether its embedded constituents correspond to inactive edges. Let us remark that in a suite, an embedded non lexical constituent is indicated by a suite of non lexical categories as in the example (4-b) indicating the embedded *AP* as a sequence of juxtaposed *AP*.

(4) a. *the very old book*

b. $\{Det_1, AP_2, AP_3, N_4\}$

When an embedded constituent corresponds to an inactive edge, then a *reduce* operation is applied, replacing the sequence of categories with a single category (in which bounds are indicated). When all embedded constituents are reduced (step 2.2.), the edge is added to the set of inactive ones.

The step 2.3. looks for juxtaposed edges. If an inactive edge can be juxtaposed to the current and if this new suite can be characterized, then the suite forms a new edge that can be added to the set of active ones.

# 5 Conclusion

Constraints offer several advantages, both for linguistics and computational reasons. Property Grammar proposes an intensive use of this notion: a grammar in this approach is an actual constraint system, which can be used as such for parsing. Moreover, PG relies on the independence between syntactic structure and the representation of syntactic relations. This specificity presents a lot of interests, in particular, the generative notion of grammaticality can be replaced with that of characterization specifying linguistic properties for any kind of input. This renders this approach intrinsically robust and, from a computational point of view, allows the use of constraints at any level of the parsing process. This last point is original and relies on the fact that a property grammar expresses constraints directly over categories which are available from the beginning of the parse whereas other approaches link constraints to local structures.

# References

[Archangeli97] Archangeli D. & D.T. Langendoen (eds.) (1997) *Optimality Theory*, Blackwell.

[Bès99] Bès G. & P. Blache (1999) “Propriétés et analyse d’un langage”, in proceedings of *TALN’99*.

[Blache99] Blache P. (1999) “Filtering and Fusion: A Technique for Parsing with Properties”, in proceedings of *NLPRS’99*.

[Blache00] Blache P. (2000) “Constraints, Linguistic Theories and Natural Language Processing”, in proceedings of *NLP-2000*.

[Borsley96] Borsley R. (1996), *Modern Phrase Structure Grammars*, Blackwell.

[Carpenter95] Carpenter B. & Penn G. (1995) “Compiling Typed Attribute-Value Logic Grammars”, in H. Bunt and M. Tomita (eds.), *Current Issues in Parsing Technologies*, Kluwer.

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

[Karlsson90] Karlsson F. (1990), “Constraint Grammar as a Framework for Parsing Running Text”, in Proceedings of *COLING’90*.

[Maruyama90] Maruyama H. (1990), “Structural Disambiguation with Constraint Propagation”, in proceedings of *COLING-ACL’98 workshop on Dependency-based Grammars*.

[Menzel98] Menzel W. & I. Schröder (1998), “Decision Procedures for Dependency Parsing Using Graded Constraints”, in proceedings of *ACL’90*.

[Meurers98] Meurers D. & Minnen G. (1998) “Off-line Constraint Propagation for Efficient HPSG Processing”, in G. Webelhuth, J.-P. Koenig, A. Kathol (eds.): *Lexical and Constructional Aspects of Linguistic Explanation*, CSLI.

[Pollard96] Pollard C. (1996), “The Nature of Constraint-Based Grammar”, PACLIC conference, reprinted in *Constructions: an HPSG Perspective*, ESSLLI’98 Lecure Notes.

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

[Samuelsson96] Samuelsson C., P. Tapainen & A. Voutilainen (1996), “Inducing Constraint Grammars”, *CLAUS Report 79*.

[Schröder00] I. Schröder, W. Menzel, K. Foth & M. Schulz (2000), “Modeling Dependency Grammars with Restricted Constraints”, submitted to the journal *TAL*.

[Shieber92] Shieber S. (1992) *Constraint-Based Grammar Formalisms*, MIT Press.