PHILIPPE BLACHE
JEAN-MARIE BALFOURIER
LPL, Université de Provence, France
E-Mail: {pb,balfourier}@lpl.univ-aix.fr

Parsing with Property Grammars

O. Introduction

The concept of constraint is of a systematic use in linguistics, as well in phonology as in syntax or semantics. All modern linguistic theories refer to it, and some, like the Optimality Theory (cf. [Archangeli97]) or HPSG (cf. [Pollard94] or [Sag99]), locate the constraints at the heart of the analysis process. Together with this concept comes the idea that all linguistic information can be represented as a set of constraints (also called a constraint system) and that the analysis can be seen as a constraints satisfaction process. Any linguistic information has a direct representation using constraints. It is thus possible to use constraints for the representation of information coming from the different levels of linguistic analysis: prosody, syntax, semantics, pragmatic, etc.

However, these ideas only are very partially implemented with classical theories. One of the main problems comes from the fact that these theories require the construction of a structure before verifying the constraint system. It is then necessary to build different structures for different level of linguistic analysis. The problem comes from the fact that such structures are not superposable (cf. [Hirst98], [Blache00b]). We propose in this paper a new approach, called Property Grammars, allowing the representation of linguistic analysis using a constraint system. We discuss more precisely the question of parsing and we propose an algorithm inspired from tabular techniques adapted for constraint processing.

1. Property Grammars

The formalism of Property Grammars 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 of organizing these properties in the form of a constraint system. Such a system plays the role of a grammar in the sense that it makes it possible to describe 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 specify 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. The following list presents these properties. This list could be supplemented according to the needs. Adding a new type of property simply requires to specify its semantics.

1. *Constituency* (noted Const): Specifies the maximal set of categories that can appear in a syntactic unit.
   Example: Const(NP) = {Det, AP, N, PP, Sup, Pro}
2. *Head* (noted Head): Specifies the possible heads. One of these categories (and only one) has to be realized.
   Example: Head(NP) = {N, Pro}
3. *Unicity* (noted Unic): Set of categories that cannot be repeated in a phrase.
   Example: Unic(NP) = {Det, N, AP, PP, Sup, Pro}
4. *Requirement* (noted □): Cooccurrency between sets of categories.
   Example: N[com] □ Det
   Exclusion (noted ≠): Cooccurrency restriction between sets of categories.
   Example: AP ≠ Sup (in a NP, a superlative cannot cooccur with an AP)
5. *Linearity* (noted <): Linear precedence constraints.
6. *Dependency* (noted □): Dependency relations between categories.

The following example gives the set of properties that can characterize the language a^nb^n:

Properties of an "S" :

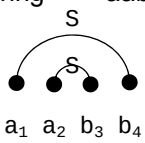
1. const = {a, b, S}
2. head = {a}
3. unic = {a, b, S}
4. a□b
5. a < b
6. S < b

This set of properties describes the category "S", the only non terminal of this grammar. The first property indicates the set of categories that can appear in the constitution of "S": one or more of these categories are possible constituents of "S". We arbitrarily chose to specify the final category "a" as a head, therefore obligatory constituent. The third property indicates the constituents which cannot be

repeated in the realization of a " S ". The requirement property (4) specifies that a "b" must be carried out at the same time as "a". The two last properties regulate the linear precedence.

We do not use in Property Grammars the derivation relation in the sense of generative grammars. This concept confers a particular statute to the hierarchical structure: it imposes the construction of a tree before being able to check its properties. In Property Grammars, each constraint can be verified separately. An utterance is grammatical (or well-formed) when it satisfies all the properties. The following example illustrates this case for the input string "aabb". In the following, we use a graph representation in which a node is a terminal and an edge represents a non-terminal. We indicate with indexes the positions of the nodes in the graph (single index for a node, double index for the source and the target of an edge).

String: "aabb":

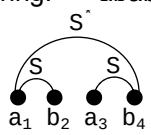


The set of categories "a_2b_3" satisfies the properties {1, 2, 3, 4, 5}. No constraint is violated. Thus, this set characterizes a category "$S_{2\text{-}3}$". The new sequence "$a_1S_{2\text{-}3}b_4$" is checked in the same way. It satisfies the properties {1, 2, 3, 4, 5, 6}. It characterizes to its turn a category "$S_{1\text{-}4}$".

In a traditional way, a sequence of categories W can correspond to a set of constituents of a higher level category C. One will say in this case that W *characterizes* C.

It is also possible to provide a precise description for utterances which do not satisfy all the properties. It is in other words a question of allowing the analysis of any kind of input, independently from any consideration on its grammaticality.

String: "abab":



The set of categories "a_1b_2" and "a_3b_4" satisfy the properties {1, 2, 3, 4, 5} and characterize an "S". At the opposite, the set of higher level categories "$S_{1\text{-}2}S_{3\text{-}4}$" violates the constraints {2, 3}. The edge "$S_{1\text{-}4}$" is noted with an asterisk in order to indicate that it does not satisfy all the properties.

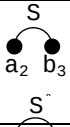
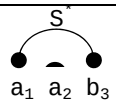
The possibility of providing indications on any type of utterance is particularly interesting for the processing of spoken languages. In this case indeed, the research of a "grammatical" structure is not relevant, such a question applying in fact for normalized uses of the language. It is preferable to provide, as proposed it Property Grammars, a *characterization* of the input to be analyzed. This characterization indicates the set of the satisfied properties and those which are not.

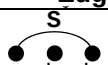
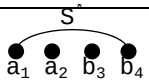
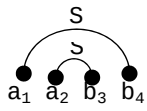
2. Chart Parsing for Property Grammars

The technique proposed here is chart-based and adapted to the processing of Property Grammars by the integration of the notion of characterization and constraint satisfaction. The principle, as for traditional chart parsers, consists in analyzing the characterization of all the sequences of nodes or edges generated during the parse. One will call hereafter *inactive edge* any edge having received a positive characterization (i.e. satisfying all the properties). The other edges are called *actives*.

The process thus consists in generating the edges (an edge is composed of a set of juxtaposed nodes or inactive edges) and in building their characterization. At the first stage, all the edges are formed by suites of nodes. When a suite corresponds to a characterization of a category *C*, the corresponding edge is labelled with *C* and stored in the stack of inactive edges (which could thus be used in the construction of new active edges). The parsing technique consists in repeating these operations until no new inactive edge can be created.

In the following example, all the active edges are associated with their characterization (i.e. the set of the satisfied properties, noted in the column P^+, and those which are not satisfied, in the column P^-). When an edge does not have any non satisfied property in its characterization, it becomes inactive.

Edge	P^+	P^-	IE
	1, 2	3, 4	
	1, 2, 3, 4, 5	∅	$S_{2\text{-}3}$
	1	2, 3	
	1, 2, 4, 5	3	

Edge	P^+	P^-	IE
	1, 2, 4, 5	3	
	1, 2, 4, 5	3	
	1, 2, 3, 4, 5, 6	∅	$S_{1\text{-}4}$

The specificity of this approach comes from the concept of inactive edge. In the previous example, we indicated that edges without non-satisfied property becomes inactive. In this case, only grammatical structures are built. But it is also possible to render inactive an edge which does not satisfy certain properties. In this case, the built structure is not grammatical in the classical sense of the term. This method allows, as for the Optimality Theory, to provide the most plausible characterization, even for non-grammatical utterances.

This parsing technique thus consists in creating a set of active edges and checking their satisfiability in the constraints system. An active edge is created when an inactive edge can replace a succession of categories of another active edge. In this case, the label of the inactive edge replaces this set of categories. It is about a traditional "reduce" operation. The algorithm schemata corresponding to this parsing technique is the following:

- *Nodes* = set of nodes
- *AE* = stack of active edges
- *IE* = stack of inactive edges
- $\wp(E)$ = set of subsets of E
- *SAT(edge)* = function calculating the characterization of "edge"
- *New* = boolean indicating a new inactive edge
- *Include(a1, a2)* = function indicating whether a sequence of categories from a2 corresponds to the edge a1.
- *Insert(a1,a2)* = function building an edge formed with a2 in which the label of the edge a1 replaces the set of corresponding categories.

1. Initialization:
   1.1. AE ← $\wp$(Nodes)
   1.2. New ← true
2. **While** (New) **do**
   2.1. ∀ edge ∈ AE
      2.1.1. ∀ i_edge ∈ IE
         2.1.1.1. **if** Include(i_edge, edge) **then**
            2.1.1.1.1. edge2 ← Insert (i_edge, edge)
            2.1.1.1.2. push(edge2, AE)
   2.2. ∀ edge ∈ AE
      2.2.1. SAT(edge)
      2.2.2. **if** P^-(edge) = ∅ **then**
         2.2.2.1. pop(edge, AE)
         2.2.2.2. push(edge, IE)

3. Example

We present in this section a detailed example illustrating the parsing technique proposed here. This example makes use of the following Property Grammar for the sentence:

Properties of the NP
(1) Const = {Det, N, AP, Sup}
(2) Oblig = {N, AP}
(3) N[com] □ Det
(4) Det < N
(5) Det < AP
(6) Det < Sup
(8) N < Sup
(9) AP ≠ Sup
(10) Det □ N
(11) AP □ N
(12) Sup □ N

Properties of the AP
(1) Const = {Adj, Adv}
(2) Oblig = {Adj}
(3) Adv < Adj
(4) Adv □ Adj

Properties of the Sup
(1) Const = {Det, Adv, Adj}
(2) Oblig = {Adj}
(3) Adj □ Det
(4) Adj □ Adv
(5) Det < Adv
(6) Det < Adv
(7) Adv < Adj
(8) Det □ Adj
(9) Adv □ Adj

Properties of the VP
(1) Const ={V, Aux, NP, AdvP, PP, AP}
(2) Oblig = {V}
(3) V < NP
(4) V < AdvP
(5) V < PP
(6) NP □ V
(7) AdvP□ V
(8) PP □ V

Properties of the S
(1) Const ={NP, VP}
(2) Oblig = {VP}
(3) NP < VP
(4) NP □ VP

Using this grammar, it is possible to characterize several syntactic units for the example:
la nuit la plus sombre a une fin lumineuse
(the darkest night has a luminous end)

The following table indicates the different inactive edges built by satisfying the properties of the grammar.

Index	Cat	Source	Target	Edges	String
1	det	1	1		la
2	N	2	2		nuit
3	det	3	3		la
4	adv	4	4		plus
5	adj	5	5		sombre
6	V	6	6		a
7	det	7	7		une
8	N	8	8		fin
9	adj	9	9		lumineuse
10	NP	1	2	det_1 N_2	la nuit
11	Sup	3	5	det_3 adv_4 adj_5	la plus sombre
12	AP	5	5	adj_5	sombre
13	VP	6	6	V_6	a
14	NP	7	8	det_7 N_8	une fin
15	AP	9	9	adj_9	lumineuse
16	NP	1	5	NP_10 Sup_11	la nuit la plus sombre
17	NP	3	5	Sup_11	la plus sombre
18	VP	6	8	V_6 NP_14	a une fin
19	NP	7	9	NP_14 AP_15	une fin lumineuse
20	P	1	6	NP_16 VP_13	la nuit la plus sombre a
21	P	1	8	NP_16 VP_18	la nuit la plus sombre a une fin
22	P	3	6	NP_17 VP_13	la plus sombre a
22	P	3	8	NP_17 VP_18	la plus sombre a une fin
24	VP	6	9	V_6 NP_19	a une fin lumineuse
25	P	1	9	NP_16 VP_24	la nuit la plus sombre a une fin lumineuse
26	P	3	9	NP_17 VP_24	la plus sombre a une fin lumineuse

4. Conclusion

The formalism of Property Grammars and the parsing technique presented here illustrate the interest of the use of the constraints for natural language processing. This approach allows a precise analysis, but can also be used in applications relying on shallow parsing which only requires checking a subset of properties (for example constituency and linear precedence). Moreover, this technique offers the advantage of being robust: one only has to indicate during the satisfaction process the number of constraints (or their type) to be satisfied. This technique thus is particularly adapted to corpus analysis and more generally to the treatment of spoken languages.

5. References

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

[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. Hirst (2000) "Multi-level annotation for spoken-language corpora", in proceedings of *ICSLP-2000*

[Hirst98] Hirst D. 1998. "Intonation in British English." in D.J.Hirst & A. Di Cristo (eds) *Intonation Systems. A Survey of Twenty Languages*. Cambridge; Cambridge University Press.

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

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

[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", *Journal TAL*.