From Shallow to Deep Parsing Using Constraint Satisfaction

Jean-Marie BALFOURIER, Philippe BLACHE & Tristan VAN RULLEN
Laboratoire Parole et Langage
29, Avenue Robert Schuman
13621 Aix-en-Provence, France
{balfourier, blache, tristan.vanrullen}@lpl.univ-aix.fr

Abstract

We present in this paper a technique allowing to choose the parsing granularity within the same approach relying on a constraint-based formalism. Its main advantage lies in the fact that the same linguistic resources are used whatever the granularity. Such a method is useful in particular for systems such as text-to-speech that usually need a simple bracketing, but in some cases requires a precise syntactic structure. We illustrate this method in comparing the results for three different granularity levels and give some figures about their respective performance in parsing a tagged corpus.

Introduction

Some NLP applications make use of shallow parsing techniques (typically the ones treating large data), some others rely on deep analysis (e.g. machine translation). The respective techniques are quite different: the former usually relies on stochastic methods where the later uses symbolic ones. However, this can constitute a problem for applications relying on shallow parsing techniques and needing in some occasions deep analysis. This is typically the case for text-to-speech systems. Such applications usually rely on shallow parsers in order to calculate intonative groups on the basis of syntactic units (or more precisely on chunks). But in some cases, such a superficial syntactic information is not precise enough. One solution would then consist in using a deep analysis for some constructions. No system exists implementing such an approach. This is in particular due to the fact that this would require two different treatments, the second one redoing the entire job. More precisely, it is difficult to imagine in the generative framework how to implement a parsing technique capable of calculating chunks and, in some cases, phrases with a possible embedded organization.

We present in this paper a formalism relying on constraints that constitutes a possible answer to this problem. This approach allows the use of a same linguistic resource (i.e. a unique grammar) that can be used fully or partially by the parser. This approach relies on the fact that (1) all linguistic information is represented by means of constraints and (2) the constraints are of regular types. The idea consists then in implementing a technique that can make use of some constraints in the case of shallow parsing, and the entire set of them for deep analysis. In our formalism, constraints are organized into different types. Tuning the granularity of the parse consists then in selecting the types of constraints to be verified.

In the first part of this paper, we present the *property grammar* formalism, its main advantages both in terms of representation and implementation. In the second part, we describe the parsing technique and the different approaches used for shallow and deep parsing. We address in particular in this section some complexity aspects illustrating the properties of the parsing techniques and we propose an evaluation over a corpus. In the third part, we illustrate the respective characteristics of the different approaches in describing for the same example the consequences of tuning the parse granularity. We conclude in presenting some perspectives for such a technique.

1 Property Grammars

The notion of constraints is of deep importance in linguistics, see for example Maruyama (1990), Pollard (1994), Sag (1999). Recent theories (from the constraint-based paradigm to the principle and parameters one) rely on this notion. One of the main interests in using constraints comes from the fact that it becomes possible to represent any kind of information

(very general as well as local or contextual one) by means of a unique device. We present in this section a formalism, called *Property Grammars*, described in Bès (1999) or Blache (2001), that makes it possible to conceive and represent all linguistic information in terms of constraints over linguistic objects. In this approach, constraints are seen as relations between two (or more) objects: it is then possible to represent information in a flat manner. The first step in this work consists in identifying the relations usually used in syntax.

This can be done empirically and we suggest, adapting a proposal from Bès (1999), the set of following constraints: *linearity, dependency, obligation, exclusion, requirement* and *uniqueness*. In a phrase-structure perspective all these constraints participate to the description of a phrase. The following figure roughly sketches their respective roles, illustrated with some examples for the *NP*.

Constraint	Definition
Linearity (<)	Linear precedence constraints
Dependency (→)	Dependency relations between categories
Obligation (*Oblig*)	Set of compulsory and unique categories. One of these categories (and only one) has to be realized in a phrase.
Exclusion ( )	Restriction of cooccurrence between sets of categories
Requirement (⇒)	Mandatory cooccurrence between sets of categories
Uniqueness (*Uniq*)	Set of categories which cannot be repeated in a phrase

In this approach, describing a phrase consists in specifying a set of constraints over some categories that can constitute it. A constraint is specified as follows. Let *R* a symbol representing a constraint relation between two (sets of) categories. A constraint of the form *a R b* stipulates that if *a* and *b* are realized, then the constraint *a R b* must be satisfied. The set of constraints describing a phrase can be represented as a graph connecting several categories.

The following example illustrates some constraints for the NP.

Linearity	*Det < N; Det < AP;* *AP < N; N < PP*
Requirement	*N[com] Þ Det*
Exclusion	*N Pro; N[prop] Det*
Dependency	*Det ® N; AP ® N; PP ® N*
Obligation	*Oblig(NP) = {N, Pro, AP}*

In this description, one can notice for example a *requirement* relation between the common noun and the determiner (such a constraint implements the complementation relation) or some *exclusion* that indicate cooccurrency restriction between a noun and a pronoun or a proper noun and a determiner. One can notice the use of sub-typing: as it is usually the case in linguistic theories, a category has several properties that can be inherited when the description of the category is refined (in our example, the type *noun* has two sub-types, *proper* and *common* represented in feature based notation). All constraints involving a noun also hold for its sub-types. Finally, the dependency relation, which is a semantic one, indicates that the dependent must combine its semantic features with the governor. In the same way as HPSG does now with the DEPS feature as described in Bouma (2001), this relation concerns any category, not necessarily the governed ones. In this way, the difference between a complement and an adjunct is that only the complement is selected by a requirement constraint, both of them being constrained with a dependency relation. This also means that a difference can be done between the syntactic head (indicated by the *oblig* constraint) and the semantic one (the governor of the dependency relation), even if in most of the cases, these categories are the same. Moreover, one can imagine the specification of dependencies within a phrase between two categories other than the head.

One of the main advantages in this approach is that constraints form a system and all constraints are at the same level. At the difference of other approaches as *Optimality Theory*, presented in Prince (1993), there exists no hierarchy between them and one can choose, according to the needs, to verify the entire set of constraints or a subpart of it. In this perspective, using a constraint satisfaction technique as basis for the parsing strategy makes it possible to implement

the possibility of verifying only a subpart of this constraint system. What is interesting is that some constraints like linearity provide indications in terms of boundaries, as described for example in Blache (1990). It follows that verifying this subset of constraints can constitute a bracketing technique. The verification of more constraints in addition to linearity allows to refine the parse. In the end, the same parsing technique (constraint satisfaction) can be used both for shallow and deep parsing. More precisely, using the same linguistic resources (lexicon and grammar), we propose a technique allowing to choose the granularity of the parse.

2 Two techniques for parsing Property Grammars

We describe in this paper different parsing techniques, from shallow to deep one, with this originality that they rely on the same formalism, described in the previous section. In other words, in our approach, one can choose the granularity level of the parse without modifying linguistic resources

2.1 Shallow parsing

In this technique, we get hierarchical and grammatical information while preserving robustness and efficiency of the processing. In this perspective, we make use of a grammar represented in the *Property Grammar* formalism described above. One of the main interests of this formalism is that it doesn't actually make use of the grammaticality notion, replacing it with a more general concept of *characterization*. A characterization simply consists in the set of the constraint system after evaluation (in other words the set of properties that are satisfied and the set of properties that are not satisfied). A characterization only formed with satisfied properties specifies a grammatical structure. In this sense, characterization subsumes grammaticality. It becomes then possible to propose a description in terms of syntactic properties for any kind of input (grammatical or not). Opening and closing chunks relies here on information compiled from the grammar. This information consists in the set of left and right potential corners, together with the potential constituents of chunks. It is obtained in compiling linear precedence, requirement and exclusion properties described in the previous sections together with, indirectly, that of constituency.

The result is a compiled grammar which is used by the parser. Two stacks, one of *opened categories* and a second of *closed categories*, are completed after the parse of each new word: we can open new categories or close already opened ones, following some rules. This algorithm being recursive, the actions *opening*, *continuing* and *closing* are recursive too. This is the reason why rules must have a strict definition in order to be sure that the algorithm is deterministic and always terminates. This shallow parsing technique can be seen as a set of production/reduction/cutting rules.

- *Rule 1*: Open a phrase *p* for the current category *c* if *c* can be the left corner of *p*.
- *Rule 2*: Do not open an already opened category if it belongs to the current phrase or is its right corner. Otherwise, we can reopen it if the current word can only be its left corner.
- *Rule 3*: Close the opened phrases if the more recently opened phrase can neither continue one of them nor be one of their right corner.
- *Rule 4*: When closing a phrase, apply rules 1, 2 and 3. This may close or open new phrases taking into consideration all phrase-level categories.

2.2 Deep parsing

Deep analysis is directly based on property grammars. It consists, for a given sentence, in building all the possible subsets of juxtaposed elements that can describe a syntactic category. A subset is positively characterized if it satisfies the constraints of a grammar. These subsets are called edges, they describe a segment of the sentence between two positions.

At the first step, each lexical category is considered as an edge of level 0. The next phase consists in producing all the possible subsets of edges at level 0. The result is a set of edges of level 1. The next steps work in the same way and produce all the possible subsets of edges, each step corresponding to a level. The algorithm ends when no new edge can be built.

An edge is characterized by:

- an initial and a final position in the sentence,

- a syntactic category,
- a set of syntactic features
- a set of constituents: a unique lexical constituent at the level 0, and one or several edges at the other levels.

After parsing, a sentence is considered as grammatical if at least one edge covering completely the sentence and labelled by the category S is produce. But even for ungrammatical cases, the set of edges represents all possible interpretations of the sentence: the set of edges contains the set of constraints that describe the input. By another way, in case of ambiguity, the parser generates several edges covering the same part and labelled with the same category. Such similar edges are distinct by their syntactical features (in the case of an ambiguity of features) or by their different constituents (typically an ambiguity of attachment).
Several heuristics allow to control the algorithm. For example, an edge at level *n* must contain at least an edge at level *n-1*. Indeed, if it would contain only edges at levels lower than *n-1*, it should have been already produced at the level *n-1*.

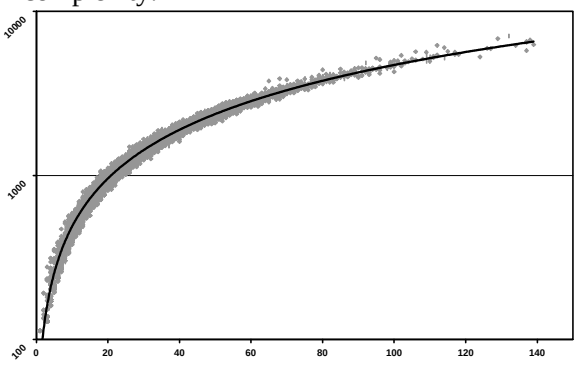
The parse ends in a finite number of steps at the following conditions:
- if the number of syntactic categories of the grammar is finite,
- if the grammar does not contain a *loop of production*. We call loop of production, the eventuality that a category c_1 can be constituted by an unique category c_2, itself constituted by an unique category c_3 and so until c_n and that one of category c_2 to c_n can be constituted by the unique category c_1.

3 Compared complexity

Of course, the difference of granularity of these algorithms does have a cost which has to be known when choosing a technique.
In order to study their complexity, we parsed a french corpus of 13,236 sentences (from the newspaper *Le Monde*), tagged by linguists (the CLIF project, headed by Talana).

3.1 Shallow parsing with Chinks and Chunks

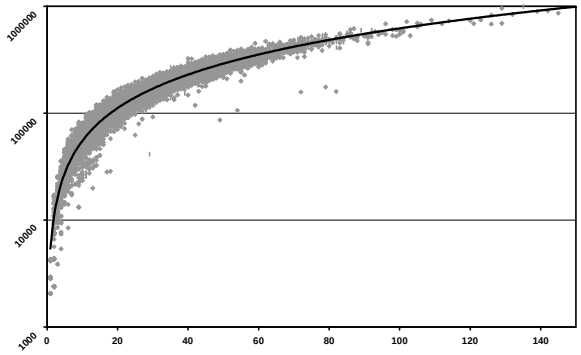
With the aim of comparing our techniques, we first built a simple robust chunker. This quick program gives an idea of a bottom complexity for the two techniques based on property grammars. This algorithm relies on the Liberman and Church's Chink&Chunk technique (see Liberman & Church (1992)) and on Di Cristo's chunker (see Di Cristo (1998) and DiCristo & al (2000)). Its mechanism consists in segmenting the input into *chunks*, by means of a finite-state automaton making use of *function words* as block borders. An improvement of the notion of chunk is implemented, using conjunctions as neutral elements for chunks being built. This algorithm constitutes an interesting (and robust) tool for example as basis for calculating prosodic units in a Text-to-Speech Synthesizer. Chink/Chunk algorithm is a simple but efficient way to detect syntactic boundaries. In the average, best and worst cases, for `M` sentences, each sentence consisting of `Nw` words, its complexity has an order of `M*Nw*Constant`. That is to say a linear complexity.



Instructions / number of words
for Chink & Chunk (logarithmic scale)

3.2 Shallow parsing with PG

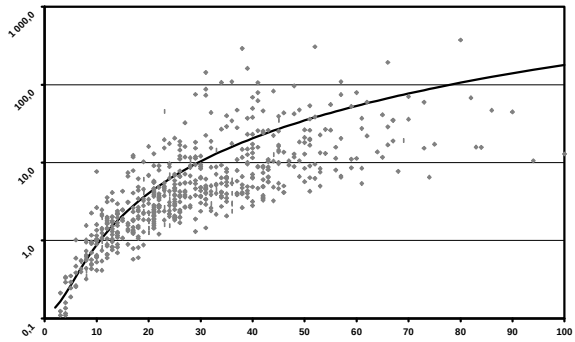
With the shallow parser algorithm, we can detect and label more syntactic and hierarchic data: in the average, worst and best cases, for `M` sentences, each sentence consisting of `Nw` words; for a set of `C` precompiled categories, its complexity has an order of `M*C*(Nw²+Nw)*Constant`. That is to say a polynomial complexity.



Instructions / number of words
for Shallow Parser (logarithmic scale)

3.3 Deep parsing with PG

For the evaluation of the deep parser algorithm, we parsed a corpora of 620 sentences of the same corpus. Unlike the two previous algorithms, the dispersal of results is much more important.



Million instructions / number of words
for Deep Parser (logarithmic scale)

In the theory, the algorithm is of exponential type but its progress is permanently constrained by the grammar. This control being heavily dependent from the grammatical context, the number of instructions necessary to parse two same size sentences can be very different. Nevertheless, in the reality of a corpus, the average complexity observed is of polynomial type. So, if `Nw` is the number of words of a sentence, the best estimate complexity of its parse corresponds to a polynomial of order 2.4 ($Nw^{2.4}$*`Constant`).

3.4 Remarks on complexity

Our study considers the parser complexity as a function of two parameters:

- the size of the parsed sentence,
- the number of grammatical categories.

This complexity is relies on the number of "simple instructions" treated by the programs. Comparing the average complexity of each parser is then a good way to know which one is faster. Ranking techniques can then be extracted from the results. It would have been interesting to compare them in terms of maximal complexity, but this is not actually possible because of an important difference between the two first parsers which are deterministic, and the last one which is not:

- for the first two techniques, the minimal, average and maximal complexities are polynomial,
- the deep parser has an exponential maximal complexity and polynomial minimal and average complexities.

Moreover, the study of the maximal complexity of the deep parser has to be treated as another problem. Usually, such a study must have to be done taking care of the size of the grammar. But with property grammars, other parameters have to be used: a property grammar is a set of constraints (Linearity, Dependency, Obligation, Exclusion, Requirement, Uniqueness) belonging to two different groups. In a formal terminology, these groups are "reduction constraints" and "increasing constraints". These groups characterize the behavior of the parser, as a formal system would do for a recognition problem. "Increasing constraints" allow the instanciation of the search space, where "reduction constraints" allow pruning this space. Most of the sentences are fastly parsed because of their well-formedness: the reduction constraints are more frequently used than increasing ones in such sentences. Ambiguous and ill-formed sentences require a greater use of increasing constraints.

Thus, the size of the grammar is less informative about the theoretical complexity than the relative importance of increasing and reduction constraints: for instance a greater grammar, with

more reduction constraints would have a lower theoretical complexity. The study of such a problem does not belong to this study because it would lead us to a different experimentation.

4 Different results

Our parsers demonstrate the possibility of a variable granularity within a same approach. We illustrate in this section the lacks and assets of the different techniques with the example below (in French):

"Le compositeur et son librettiste ont su créer un équilibre dramatique astucieux en mariant la comédie espiègle voire égrillarde et le drame le plus profond au cœur des mêmes personnages."

"*The composer and his librettist successfully introduced an astute dramatic balance in marrying the mischievous, ribald comedy with the deepest drama for the same characters*."

4.1 Chink/chunk approach

```
[(sentence)
  [(chunk)Le compositeur et son librettiste
ont su créer]
  [(chunk)un équilibre dramatique astucieux]
  [(chunk)en mariant]
  [(chunk)la comédie espiègle]
  [(chunk)voire égrillarde]
  [(chunk)et le drame]
  [(chunk)le plus profond]
  [(chunk)au coeur des mêmes personnages]]
```

This first example shows a non-hierarchical representation of the sentence, divided into chunks. No linguistic information is given.

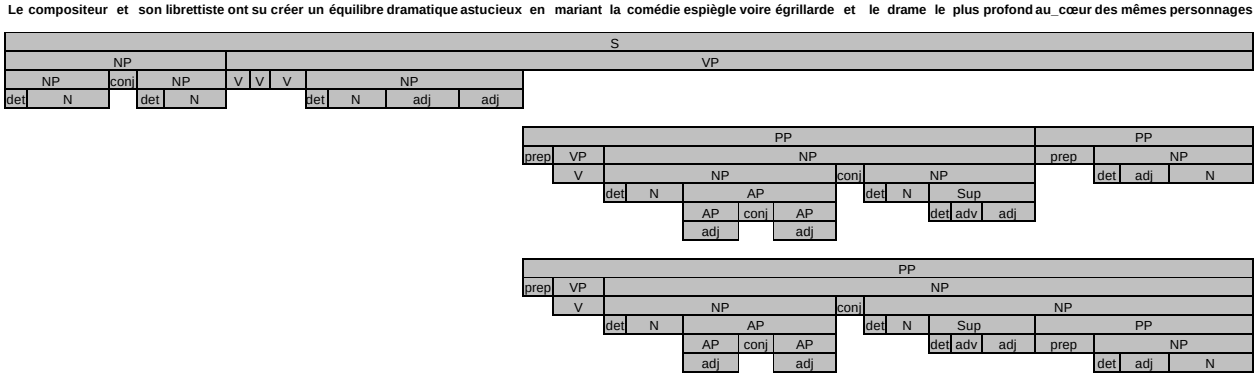
4.2 Shallow parsing approach

```
[(sentence)
  [(NP)Le compositeur
    [(AP) et]
    son librettiste]
  [(VP)ont su créer]
  [(NP) un équilibre
    [(AP)dramatique astucieux]]
  [(Compl)en
    [(VP)mariant]]
  [(NP)la comédie
    [(AP)espiègle voire égrillarde et]
    le drame
    [(Sup)le plus profond]]
  [(PP)au cœur de
    [(NP)les
      [(AP)mêmes]
      personnages]]]
```

This second example gives a hierarchical representation of the sentence, divided into grammatically tagged chunks. Because we used a precompiled version of the grammar (shortened) and because we forced some syntactic choices in order to keep a determinist and finishing parsing, it appears that some errors have been made by the shallow parser: Conjunctions are (badly) distinguished as Adverbial Phrases. In spite of these gaps, cutting is improved and most of the categories are detected correctly.

4.3 Deep parsing approach

The last example (next figure) presents two of the maximum coverages produced by the deep parser. This figure, which illustrates the PP attachment ambiguity, only presents for readabiulity reasons the hierarchical structure. However, remind the fact that each label represents in fact a description which the state of the constraint system after evaluation.



5 Conclusion

The experiments presented in this paper show that it is possible to calculate efficiently the different kind of syntactic structures of a sentence using the same linguistic resources. Moreover, the constraint-based framework proposed here makes it possible to choose the granularity, from a rough boundary detection to a deep non-deterministic analysis, via a shallow and deterministic one. The possibility of selecting a granularity level according to the data to be parsed or to the targetted application is then very useful.

An interesting result for further studies lies in the perspective of combining or multiplexing different approaches. It is for example interesting to notice that common boundaries obtained by these algorithms eliminates ill-formed and least remarkable boundaries. At the same time, it increases the size of the blocks while maintaining the linguistic information available (this remains one of the most important problems for text-to-speech systems). Finally, it allows to propose a parameterized granularity in balancing the relative importance of different competing approaches.

References

Abney, S. (1991) "Parsing by chunks"., in Berwick, R., Abney, S., Tenny, C. (eds.). *Principle-based Parsing*, Kluwer Academic Publishers, 257-278.

Abney S. (1996) "Partial Parsing via Finite-State Calculus", in proceedings of *ESSLLI'96 Robust Parsing Workshop*.

Abney, S. (1997) "Part-of-speech tagging and partial parsing", in Young, S., Bloothooft, G. *Corpus-Based Methods in Language and Speech Processing*, Kluwer Academic Publishers, 118-136.

Allen, J., Hunnincutt, S., Carlson, R., Granström, B. (1979) "MITalk-79 : The 1979 MIT text-to-speech system", in Wolf and Klatt (eds.), *Speech Communications*, Papers Presented at the 97th Meeting of the ASA: 507-510.

Allen, J., Hunnincutt, S., Klatt, D. (1987) "From text to speech: The MITalk system", Cambridge University Press.

Bès G. & P. Blache (1999) "Propriétés et analyse d'un langage", in proceedings of *TALN'99*.

Blache P. & J.-Y. Morin (1990) "Bottom-up Filtering: a Parsing Strategy for GPSG", in proceedings of *COLING'90*.

Blache P. & J.-M. Balfourier (2001) "Property Grammars: a Flexible Constraint-Based Approach to Parsing", in proceedings of *IWPT-2001*.

Bouma G., R. Malouf & I. Sag (2001) "Satisfying Constraints on Extraction and Adjunction", in *Natural Language and Linguistic Theory*, 19:1, Kluwer.

Chanod J.-P. (2000) "Robust Parsing and Beyond", in Robustness in Language Technology, Kluwer.

Di Cristo P, (1998). Génération automatique de la prosodie pour la synthèse à partir du texte. Ph.D. thesis, Université de Provence, France.

Di Cristo A., Di Cristo P., Campione E, Veronis J, (2000). A prosodic model for text to speech synthesis in French.

Duchier D. & R. Debusmann (2001) "Topological Dependency Trees: A Constraint-Based Account of Linear Precedence", in proceedings of *ACL*.

Grinberg D., J. Lafferty & D. Sleator (1995), A robust parsing algorithm for link grammars, CMU-CS-95-125, Carnegie Mellon University.

Kübler S. & E. Hinrichs (2001) "From Chunks to Function-Argument Structure: A similarity-Based Approach", in proceedings of *ACL-01*.

Liberman, M., Church, K. (1992) "Text analysis and word pronunciation in text-to- speech synthesis", in Furui, S., Sondhi, M.M. (eds), *Advances in Speech Signal Processing*, Dekker, 791-831.

Maruyama H. (1990), "Structural Disambiguation with Constraint Propagation", in proceedings of *ACL'90*.

Pollard C. & I. Sag (1994), *Head-driven Phrase Structure Grammars*, CSLI, Chicago University Press.

Prince A. & P. Smolensky (1993) *Optimality Theory: Constraint Interaction in Generative Grammars*, Technical Report RUCCS TR2, Rutgers Center for Cognitive Science.s

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