Extracting Selected Phrases through Constraint Satisfaction

Veronica Dahl, Philippe Blache

Extracting Selected Phrases through Constraint Satisfaction

Veronica Dahl[1] and Philippe Blache[2]

[1] Simon Fraser University
Vancouver, Canada
veronica@cs.sfu.ca
[2] LPL, Université de Provence
Aix-en-Provence, France
pb@lpl.univ-aix.fr

Abstract. We present in this paper a CHR based parsing methodology for parsing *Property Grammars*. This approach constitutes a flexible parsing technology in which the notions of derivation and hierarchy give way to the more flexible notion of constraint satisfaction between categories. It becomes then possible to describe the syntactic characteristics of a category in terms of satisfied and violated constraints.
Different applications can take advantage of such flexibility, in particular in the case where information comes from part of the input and requires the identification of selected phrases such as *NP, PP*, etc. Our method presents two main advantages: first, there is no need to build an entire syntactic structure, only the selected phrases can be extracted. Moreover, such extraction can be done even from incomplete or erroneous text: indication of possible kinds of error or incompleteness can be given together with the proposed analysis for the phrases being sought.

1 Introduction

Extracting selected phrases from written or spoken text is an important step for many useful applications of language processing. For instance, concept extraction or text summarization can benefit from a preliminary identification of all noun phrases or verbs, to be further processed, e.g. through consulting a concept hierarchy; question answering can focus on noun phrases or verbs at least for the replies to be given; command oriented systems might focus on verb phrases; temporal systems, on time adverbial phrases, etc.

Ideally we want to be able to extract concepts from text produced in real life conversation, which typically is incomplete, often not perfectly grammatical, and sometimes erroneous. Imperfections can result from normal human error in actual speech, or be introduced by machines, as in the case of text produced from speech recognition systems, which are renowned for their error-proneness.

Flexible parsing techniques offer great advantages in this perspective. Among possible solutions, constraint-based approaches allow us to use the same parsing mechanism (constraint satisfaction) whether the grammar is incomplete, heterogeneous, etc. Moreover, provided that no extra mechanisms than satisfaction are used, constraints can be relaxed in accordance to user-defined criteria. In particular, this can be done selectively (for some sentences but not others). For instance, we may want to express that a noun requires a determiner inside a noun phrase, unless we are dealing with a generic statement, as in "Lions sleep". To this effect, we can test the conditions relevant to genericity in the body of the rule that relaxes the constraint imposing a determiner, so that only upon them being satisfied will the constraint be relaxed.

Property-based linguistic models (see [Bès99a,Bès99b], [Blache05]) view linguistic constraints as properties between sets of categories, rather than in the more traditional terms of properties on hierarchical representations of completely parsed sentences. This view has several advantages, such as allowing for mistakes to be detected and pointed out rather than blocking the analysis altogether, and facilitates dynamic processing of text produced on the fly, as needed for the growing number of applications involving speech.

In this paper we refine a methodology for *Property Grammars* (cf. [Blache05]) first proposed in [Dahl04b] which relies exclusively on constraints. It controls the parse through head-driven analysis, provides a direct interpretation of this formalism while preserving all its theoretical properties at the implementation level, and can focus on specific phrases- in the case of our toy implementation, noun phrases- as mandated by a user's modular and easy to change command. As the implementation language, we use a specialized grammatical formalism called CHRG (Constraint Handling Rule Grammars) described in [Christiansen01] on top of CHR [Frühwirth98].

2 Representing syntax with constraints

The basic idea of *Property Grammars* is to represent different kinds of syntactic information separately. In this approach, syntactic structure is not expressed in terms of hierarchy, but only by means of relations between categories. We describe in this section how to represent such relations in terms of constraints and take advantage of this in the perspective of a direct constraint-based implementation.

2.1 Background: Property Grammars

In our approach, syntactic properties rely on relations that do not have specific topological constraints (they can for example be crossed). Categories are described by means of such relations. As a consequence, the notion of constituency is no longer crucial for the description process: a category is specified by a set of properties rather than by a set of constituents. The fact that several categories belong to a network of relations indicates that they characterize an upper-level

category. A syntactic category is then described by a set of properties which represent relations between other categories (lexical or syntactic).

In this approach, the goal is to make explicit all the different relations that can exist. We distinguish in this perspective the following types of information:

- linear precedence, which is an order relation between categories,
- subcategorization, which indicates coocurrence relations between categories or sets of categories,
- the impossibility of cooccurrence between categories,
- the impossibility for a category to be repeated,
- the minimal set of obligatory constituents (usually one single constituent) which is the head,
- semantic relations between categories, in terms of dependency.

These different kinds of information correspond to different properties, respectively: *linearity, requirement, exclusion, unicity, obligation, dependency.* Such information can always be expressed in terms of relations between categories, as shown in the following examples:

- Linear precedence: $Det \prec N$ (a determiner precedes the noun)
- Dependency: $AP \rightsquigarrow N$ (an adjectival phrase depends on the noun)
- Requirement: $V[inf] \Rightarrow to$ (an infinitive comes with *to*)
- Exclusion: $seems \not\Rightarrow ThatClause[subj]$ (the verb *seems* cannot have *That* clause subjects)

All syntactic categories are characterized by a set of relations that forms a connected graph. The syntactic description of a language consists of all the different relations that can be expressed between categories. A relation (also called a property) can be conceived as a constraint on the set of categories. A grammar is then a set of constraints and satisfiability becomes the core of the parsing process (see [Blache01]). What is interesting in this approach is that no implicit information, for example under the form of a specific mechanism, is needed. In particular, there is no need to build a structure before being able to verify its properties as it is the case with classical generative approaches (although for convenience given what researchers are used to, we do provide a tree as well as a side effect of parsing). Moreover, using satisfiability alone has important consequences in the conception of the syntactic structure. Evaluating a constraint system for a given set of categories allows us to specify precisely the set of properties that are verified. In the same way, in the case of ill-formed input, such evaluation identifies precisely the set of satisfied and violated constraints. Such a result is then of deep interest in the sense that it identifies precisely all the specificities of an input. In *Property Grammars* this information constitutes the output of a parse, which is but the status of the constraint system after evaluation.

2.2 The parsing schema

The basic mechanism in constraint satisfaction problems is to find, for a given set of variables, an assignment that satisfies the constraint system. In the problem addressed here, the variables are taken from the set of categories. An assignment is given from an input (i.e. the sentence to be parsed). Starting from the set of lexical categories corresponding to the words of the sentence, all possible assignments (i.e. subsets of categories) are evaluated. When a syntactic category is characterized, it is added to the set of categories to be evaluated. This approach is basically incremental in the sense that any subset of categories can be evaluated. This means that an assignment A can be completed by adding other categories. When syntactic categories are inferred after the first step of the process, it is then possible to complete the first assignments (made with lexical categories) with new syntactic ones.

We have seen that in Property Grammars, a category is described by a set of constraints. But reciprocally, it is possible to identify a category from a given property. This is typically the case with properties expressing relationships between categories, such as linearity, requirement, obligation and dependency. Evaluating such properties makes it possible to infer that the syntactic category to which the property is attached is being characterized. We have referred to these properties under the collective name of *selection* constraints.

The role of selection constraints is central to our approach. The reason such constraints allow us to select the characterized category is that they are local to this category. Moreover, in some cases they have a global scope over the category : their satisfiability value (i.e. satisfied or violated) cannot change for a given category whatever the subset of constituents. As soon as the constraint can be evaluated, this value is permanent. For example, when a linearity or a dependency constraint is satisfied, adding new constituents to the category cannot change this fact. Other kinds of constraints have to be re-evaluated at each stage. For instance, when adding a new category, we need to verify that unicity and exclusion are still satisfied. In all that follows, we call the latter *filtering constraints*. Contrary to *selection constraints*, one cannot infer the materialization of a syntactic category from their evaluation. They play a filtering role in the sense that they rule out some construction. Yet another type of constraint, which we call *recoverable constraints* can succeed by the incorporation of one more category into a given phrase for which, without this added category, the constraint failed.

Let us examine what the consequences are on constraint evaluation. As explained above, the principle consists in completing original assignments with new categories when they are inferred. Insofar as the evaluation of selection constraints (as soon as this evaluation can be performed) is valid through a complete assignment, whatever its constituents, it is not necessary to re-calculate it. In other words, when an assignment A is made by completing another assignment A', the set of selection constraints of A' is inherited by A. We describe in the next section the different types of properties and their consequence for new assign-

ments in more detail, with respect to a specific instance of the general parsing schema we present here.

In the following, we note selection and filtering constraints as $R_{select}(C, XP)$ and $R_{filter}(C, XP)$ in which C is the constraint and XP the syntactic category to which the constraint is associated. For some assignment A, a constraint is relevant (or can be evaluated) when the categories of A are a subset of the categories involved in C. We note the fact that A can be evaluated for some constraint as follows: $A/R_{select}(C, XP)$. We note the set of filtering and selection constraints for a given category XP by $R(C, XP) = R_{select}(C, XP) \cup R_{filter}(C, XP)$. Finally, we note the state of the constraint system Σ for an assignment A after evaluation by $SAT(A, \Sigma)$. Each category is indexed by its boundaries, noted $c_{(i,j)}$.

Let K be the set of categories, noted c_i, let i, j, k some indexes such that $i < j < k$:

1. $A \leftarrow \{c_i, ..., c_j\}$
2. if $A/R_{select}(C, XP)$
3. instantiate $XP_{(i,j)}$
4. $\Sigma = \bigcup R(C, XP)$
5. $Char(A) \leftarrow SAT(A, \Sigma)$
6. while $SAT(A, \Sigma)$ acceptable
7. $A \leftarrow A \cup \{c_k\}$
8. $\Sigma = \bigcup R(C, XP)$
9. $Char(A) \leftarrow SAT(A, \Sigma)$
10. $k \leftarrow k + 1$

In this algorithm schema, the mechanism consists in evaluating the characterization of all sequences of categories. The specificity of selection constraints is used as a control device: when a selection constraint is satisfied, the described category XP is instantiated and the related set of constraints Σ is activated. It is interesting to notice that a syntactic category can be projected by any selection constraint, independently from any constituency information (especially the head). Each new assignment, built by adding new juxtaposed categories to the initial set, is then evaluated. Such a completion of the initial assignment is possible when the satisfiability of Σ for this new assignment is *acceptable* (cf. line 7). This notion implements the parser flexibility. When we need to build only grammatical structures, *acceptability* is reduced to satisfiability. But for more flexible parsing needs (e.g. spoken language), constraints can be relaxed. The set of constraints to be relaxed, their number, etc. is indicated at this point. Finally, the general process is repeated until no new category can be added.

This parsing schema proposes a general framework in which constraints can be integrated. Each property is implemented by a constraint solver. The mechanism consists in building a characterization for each possible assignment (i.e. any subset of categories). The particularity of selection constraints plays an important role in this schema. In a classical bottom-up technique, the mechanism consists in finding a *handle* which links a set of categories with a non-terminal.

Such a relation in our approach is established between a set of properties and a category. In contrast with phrase-structure techniques, the notion of constituency does not play any particular role. As soon as a selection constraint is evaluated, the corresponding syntactic category is added to the set of categories and all the constraints participating in its description are activated. Concretely, all the selection constraints can be evaluated with no need to know the upper-level category, in contrast to filtering constraints which have to be activated.

Different strategies can be applied according to the needs of the parse. A restricted application stipulates that all constraints have to be satisfied. In this case, only grammatical characterization are built, all ill-formed structures are ruled out. For more flexible applications, typically in the case of parsing spoken language material, constraints have to be relaxed. In this case, characterizations can contain violated constraints. We next describe how CFGs are used in our approach to achieve direct interpretation in these kinds of flexible applications.

3 Direct interpretation in CHR

Our methodology for Property Grammar parsing has been designed to provide direct interpretation of Property Grammar rules. This is an interesting contribution with respect to Property Grammars themselves, but also a novel and important proof-of-concept that can lead the way for any constraint-based parsing formalism which relates categories contextually through their properties. To the best of our knowledge, our methodology is the first one that permits an expression of such parsing constraints which is directly and efficiently executable. Thus, we can say that our approach represents for grammars based on contextual properties what DCGs represent for context-free grammars, in the sense that they are as directly executable descriptive formalisms as DCGs[3].

In this section we describe the different components of our methodology : the notion of extended categories, which includes not only traditional information such as category names and features, but also the category's characterization in terms of satisfied and unsatisfied constraints; the modular notation through which a user defines the properties of a given grammar, the single rule through which parsing proceeds, and the analysis of property inheritance which is used in our system.

3.1 Extended Categories

Extended categories are of the form: *cat(Name, Features, Graph, Sat, Unsat)* where *Name* is the category name, *Features* a list of features associated with the category (which may be used to check some of the properties between categories), *Graph* is a parse tree which is obtained as a side effect of parsing (which is built

[3] Of course, DCGs do permit context sensitive parsing as well, but the context sensitivity cannot be directly expressed through symbol contiguity, it has to be indirectly expressed in extra arguments or through other extra devices such as linear implication.

even in those cases of incorrect input), and *Sat* and *Unsat* are respectively, the list of satisfied and unsatisfied properties that the immediate daughters of *Name* inside the *Graph* verify between them. In the case of single word categories, the *Sat* and *Unsat* lists will be empty. These categories are created automatically from user's lexical definitions, which are done in terms of CHRG. For instance, a user's entry:

(1) `[the] ::> cat(det,[singular,masculin]).`

compiles into:

(2) `[the] ::> cat(det,[singular,masculin],det(the),[],[]).`

Because these are CHRG rules (i.e., grammar rules, as opposed to plain CHR rules), word boundaries are carried invisibly. If needed, we can retrieve them in a grammar rule by adding `:(Start,End)` after the category, which will unify `Start` to the starting point of the category, and `End` to its end point, or we can write a plain CHR rule that looks at `cat/5` not as a grammar rule, but as the CHR constraint it compiles into, in which case `Start` and `End` can be retrieved as the two first arguments of the corresponding constraint, `cat/7`.

3.2 User defined properties

Our system allows the user to enter the specific linear precedence, dependency, requirement, exclusion, constituency and unicity properties that apply to the grammar being defined, through simple primitive predicates which are respectively `prec/3,dep/3,req/3, exclude/3, cons/2`, and `one/2`. Figure 3.2 exemplifies for a simplified noun phrase.

Linear precedence	*Dependence*	*Constituency*

`prec(det,n,sn).`
`prec(det,sa,sn).`
`prec(n,sa,sn).`
	`dep(det,n,sn).`
	`dep(n,sa,sn).`
		`cons(sn,[det,adj,sa,n]).`
		`cons(sa,adj]).`

Unicity	*Exclusion*	*Requirement*	*Phrases*
`one(det,sn).`	`exclude(sa,sup,sn).`	`req(n,det,sn).`	`xp(sn).` `xp(sa).`

Fig. 1. User defined properties

It is to be noted that while the user's definition of constituency is not rigorously needed (since, as we have seen, when selection properties are verified, the determination of constituency follows as a side effect), having it explicitly defined results in improved efficiency. Likewise, phrase definitions can be inferred from any of the other properties, but defining them explicitly makes the system easier and more readable. The user's definitions of properties will be called from system predicates which verify each of these properties on a given set of categories, as we shall see next.

3.3 A single rule for inferring all new categories

Conceptually, little more than a single rule is enough for a string's complete bottom-up parse from contiguous constituents. This rule combines two consecutive categories (one of which is of type `XP` or obligatory) into a third, by testing each of the properties on the pair and creating the new property lists through property inheritance (cf. next section). Its form is described in Fig. 2.

```
cat(X,Y,L,TL,RL,SL,UL), cat(Y,Z,R,TR,RR,SR,UR) ==>
    (k_or_xp(R,XP) -> (Ext=L, TE=TR, RE=RR, Sat0=SR, Unsat0=UR);
    (k_or_xp(L,XP) -> (Ext=R, TE=TL, RE=RL, Sat0=SL, Unsat0=UL); fail) ),
    !, ok_in(XP,Ext),
    sat_properties(L,TL,RL,R,TR,RR,XP,Sat0,Unsat0,T,Sat,Unsat)
    | cat(X,Z,XP,TE,T,Sat,Unsat).
```

Fig. 2. New Category Inference

This rule tests that one of the two categories (left or right) is a kernel (a phrase head) and the other one of its extension (i.e., complement or adjunct), and then assigns the corresponding features. It then successively tests each of the PG properties among those categories, incrementally building as it goes along the lists of satisfied and unsatisfied properties. Finally, it infers a new category of type `XP` spanning both these categories, with the finally obtained `Sat` and `Unsat` lists as its characterization. In practice, this rule unfolds into two symmetric parts, to accommodate the situation in which the XP category appears before the category Cat which is to be incorporated into it.

Constituents with discontinuities must also be allowed, for completeness. In this case we consider two categories where the end node of the first does not coincide with the start node of the second. Our current research does not yet include discontinuous constituents, for which further linguistic constraints for avoiding combinatorial explosion need to be incorporated.

The parser here described includes as well a sophisticated analysis of types of properties which helps determine how the properties that have been found to hold (fail) in a given phrase are inherited or not by the new phrase comprising that phrase plus a category being incorporated into it. Knowing whether a given property needs to be simply inherited (in which case it just remains as is in the new list of properties) or affects previously recognized properties, and how, allows us an efficient way to move from one list of properties to the next. This analysis is beyond the scope of the present paper. Complete details can be found in [Dahl04b]. The same work proposes a detailed analysis on how to refine the notion of characterization. For our purposes here, let us just say that the user can declare certain properties as relaxed, which results in sentences containing errors related to those properties to be accepted nevertheless, while indication of their failure is given in the output (in the form of a list of unsatisfied properties).

4 Extracting selected phrases as a side effect of parsing

One of the interests of our approach lies in its flexibility. The kind of parsing techniques presented above makes it possible to parse substructures as well as subpart of the input. This functionality is useful to deal with part of the input for which no information can be built. It is for example the case with possibly long lists of juxtaposed *NP*, frequent in spoken languages but for which no specific syntactic relation can be given. But it is also interesting for some applications in which the entire syntactic structure or, in other words, the description of all possible syntactic categories is necessary. This is the case for question-answering, information extraction, or terminological applications based on *NP* recognition. In these systems, the knowledge of the argument structure, or the precise description of the relations between the different categories is not necessary. We basically need to know what are the *NP*s of the input and, if possible, their contents. Several techniques can be applied for this (see [Osborne99], [Tjong00]) that usually rely on the specification of patterns defining base *NP*s. In such approaches, it is difficult or even impossible to identify the kind of relations that links the different constituents of the NP. In other words, it is not possible to choose the granularity level. In *PG*, this possibility exists, simply by stipulating the constraints that have to be satisfied among the entire constraint system that forms the grammar. This means that any category can be extracted. Moreover, the description granularity of the extracted category can also be chosen, according to the needs.

4.1 Implementation details

Concretely, all irrelevant information is simply not taken into account. For example, the categories that do not participate to a *NP* are ignored. The only special mechanism needed to skip them is a filtering set of predicates which ensures that every category different from *NP* is deleted from the constraint store upon the parse having finished, as first proposed in [Christiansen05a]. For the case of noun phrases, we need to define a constraint we will call cleanup, which we will call after the analysis of any phrase, and, for every category X different from an *NP*, a simpagation rule of the form:

(3) $\texttt{cat}(\texttt{X}, \_, \_, \_, \_), !\texttt{cleanup} <:> \texttt{true}.$

The exclamation mark combined with the rewrite symbol shown indicate sympagation in CHRG: once the parse is completed and cleanup is put in the store, the above rule removes from the store the matching category, and the constraint cleanup remains for further use, in order to remove all categories different from np.

The generation of such rules can be automated from a user's command to declare what phrases to focus on, e.g.:

(4) $:-\texttt{focus}(\texttt{np}).$

Similarly, we can include a mechanism for only leaving the outermost *NP*s in the case of embedded ones. This option can be specified by the user through the command

(5) `:−outermost(np).`

It is also possible to imagine an easy reuse from a language to another, simply by adapting the properties. In the case of basic *NP*s, very close properties are used for French and English.

4.2 Experiment

We did some experiments in applying this technique to a French medical corpus. The following example illustrates the identification of some noun phrases, extracted together with their syntactic characterizations. In these examples, the output contains the type of the phrase (in this experiment, the noun phrase), its morphosyntactic properties, its constituents and its characterization. This last part of the output, as explained in the second section, is formed by the set of satisfied and violated properties.

The example (6) illustrates the case of a simple *NP*, formed with a determiner and a noun. Its characterization shows for example the satisfaction of a precedence constraint between the determiner and the noun, some uniqueness relations for these constituents as well as a mandatory cooccurrency between the determiner and the noun.

(6)

Input	les cellules (*the cells*)

Output	`cat(np,[plu,masc], sn(det(les),n(cellules)),`
	`[prec(det,n), dep(det,n), unicity(det), unicity(n),`
	`exige(n,det), exclude(name,det), exclude(name,n)],[])`

The second example, presented in (7), shows the integration of an embedded *AP*. In this example, the *AP* (composed with a single adjective) has been identified separately. The characterization of the *NP* works exactly in the same way as before (as explained in the presentation of the algorithm). In this example too, all constraints belonging to the characterization are satisfied: precedence, exclusion, unicity, etc. One can also remark the stipulation of a dependency relation between the AP and the noun, that illustrates the capacity of the technique to identify syntactico-semantic relations.

(7)

Input	les meilleures stratégies (*the best strategies*)

Output	`cat(np,[plu,masc], np(det(les), ap(adj(meilleures)),`
	`n(stratégies)), [dep(ap,n), unicity(n),`
	`exclude(name,n), exclude(name,ap), exclude(ap,sup),`
	`prec(det,n), dep(det,n), unicity(det), exige(n,det),`
	`exclude(name,det)],[])`

The example (8) presents the case of a complex *NP*. More precisely, this is the case of an ill-formed input, due to a wrong POS-tagging: in this case, the last

adjectives of the list have been tagged as noun instead of adjectives. However, the *NP* has been recognized, due to the possibility of relaxing constraints. In this case, the uniqueness constraint applied to the noun has been violated. This constraint belongs to the second part of the characterization in a separate list of violated properties.

(8)

Input	les cellules endothéliales immunotoxines peptides proapoptotiques (*the endothelials ... cells*)
Output	`cat(np,[sing,masc],sn(det(les), n(cellules), ap(adj(endothéliales)), n(immunotoxines), n(peptides), n(proapoptotiques)), [prec(det,n), dep(det,n), exige(n,det), exclude(name,det), exclude(name,n), dep(sa,n), exclude(name,sa), exclude(sa,sup)], [unicity(n)])`

4.3 Towards semantics

Using such a symbolic robust approach to *XP* extraction makes it possible to consider integrating semantic information. This is a great advantage in comparison to other methods. Some dependencies can be included in the description so that the extraction can come with different kinds of information, for example concerning the argument structure or the roles of the possible arguments.

We develop in the following the example of *NP* extraction, illustrating how such task is implemented in order to identify *NP*s together with their main modifiers. Basically, the kind of information that is needed in applications mentioned above concerns the noun itself plus its different modifications. This means its quantification (if any) and its classical direct modifiers. In the system described here, the semantic structure of a *NP* is described by its head (the noun itself), its specifier (the determiner) and two possible modifiers (*AP* and *PP*).

(9)
$$\text{SEM} \begin{bmatrix} \text{HEAD } N \\ \text{SPEC } Det \\ \text{MOD } AP \\ \text{COMP } PP \end{bmatrix}$$

Building such structure from a *PG* parser is direct. A minimal set of properties has to be checked in order to identify the *NP* and build the structure. In order to simplify the description and improve the efficiency of the system, we do not take into account relative clauses here. Moreover, also in a simplification perspective, we do not build embedded phrases. The only authorized one is the NP itself. In the end, we obtain a description of the NP which is more complete than the classical definition of a "base NP" (see [Osborne99]) and moreover makes it possible to directly identify the relations inside the *NP*. The properties are then the following:

Constituency	Const = {N, Det, Adv, Adj, Prep, NP}
Linearity	Det $\prec$ Adv; Det $\prec$ N; Adv $\prec$ Adj; Adj $\prec$ N; N $\prec$ Prep; Prep $\prec$ NP
Requirement	Adv $\Rightarrow$ Adj; NP $\Rightarrow$ Prep

In *PG*, the entire representation of an objects (see [Blache05]) contains on the one hand its properties, and on the other hand, its local information represented in terms of features. The final description of the *NP* we use in the perspective of the *NP* extraction system is then:

(10)

NP	
FORM	HEAD *N* SPEC *Det* MOD *Adj* COMP *Prep*
PROPERTIES	Const = {Det,N,Adj,Prep, Adv, NP} Det ≺ Adv; Det ≺ N; Adv ≺ Adj; etc. Adv ⇒ Adj; NP ⇒ Prep

The extraction mechanism consists then in satisfying the set of constraints defined in the property part of the NP object. When, a sequence of words from the input satisfies this set of constraints, the corresponding structure (called FORM) is built. By means of unification, the respective arguments of the structure will be instantiated with the desired values.

5 Discussion, Conclusion

In our approach, as we have seen, syntactic categories are inferred from the evaluation of properties, without any need of constituency information.

This aspect has important consequences on the role of constraints in the parsing process. One of the problems with constraint-based approaches is that constraints are usually expressed over high-level objects or structures. This is the case for example in HPSG, in which complex feature-structures must first be built before constraints can be evaluated. Similarly, Optimality Theory also generates a set of structures (or candidate structures) and then uses constraints to filter this set. In our approach, any constraint can be evaluated at any time for any set of categories. Such evaluation, as explained above, dynamically adds new information: the satisfaction of a *selection* constraint instantiates the syntactic category it describes. But this instantiation is conceived almost as a side effect of evaluation: satisfying constraints does not rely on the knowledge of the upper-level category. In other words, the hierarchical information is no longer preponderant in the parsing process. This means that one can evaluate subsets of constraints, for example in the case of applications that only need NP recognition. In this approach, the conception of the relationship between grammar and language becomes very different from that of the generative paradigm. In the latter, a language is conceived as being generated by a grammar. In Property Grammars, a grammar is only used as a characterization device of the language properties.

As a consequence, instead of restricting the role of parsing to the evaluation of the input's grammaticality, we can propose a more flexible vision, in which a

parser's output is the description of all the properties of the input. Concretely, such a description consists in the state of the constraint system after evaluation– in other words, the set of satisfied and violated constraints. We call such state a *characterization* of the input. In some cases, a characterization only contains satisfied constraints, but it can also be the case that some constraints can be violated, especially when parsing real life corpora. In most cases, such violations do not have consequences on the acceptability of the input.

One other formalism that shares the aims and some of the features of Property Grammars are Dependency Grammars (cf. [Tesnière59] and on this point [Mel'čuk88]), a purely equational system in which the notion of generation, or derivation between an abstract structure and a given string, is also absent. However, whereas in Dependency Grammars, as their name indicates, the property of dependence plays a fundamental role, in the framework we are considering it is but one of the many properties contributing to a category's characterization. Perhaps the work that most relates to ours is Morawietz's [Morawietz00], which implements deductive parsing [Shieber95] in CHR, and proposes different types of parsing strategies (including one for Property Grammars) as specializations of a general bottom-up parser. Efficiency however is not addressed beyond a general discussion of possible improvements, so while theoretically interesting, this methodology is in practice unusable due to combinatorial explosion. Moreover, it produces all properties that apply for each pair of categories without keeping track of how these categories are formed in terms of their subcategories, so there is no easy way to make sense of the output in terms of a complete analysis of a given input string.

The idea of throwing away the traditional, hierarchical parsing scheme in favour of a view of parsing which involves properties on categories rather than rewriting schemes first materialized in the 5P formalism (cf. [Bès99a,Bès99b]). Preliminary work re. the advisability of a direct implementation of such an approach had yielded pessimistic results : [Blache95] showed that the mechanism of verification of a constraint system for syntactic analysis could be very expensive, given that the satisfiability of the system had to be verified in each stage. In the present work, however, we have moved beyond that obstacle by our analysis of property inheritance, which removes the need to recalculate all properties at each stage, allowing us to inherit at each stage most of the previous stage's properties, while calculating only the minimally necessary new properties and updating the previous properties along the lines of our property inheritance analysis. Thus, our work has validated the model of property-centered parsing with respect to efficiency, while preserving the level of generality of this theory. In addition, a direct interpretation guarantees a better evolution of the initial system: it can better adjust to changes in the theory and to experimental stages.

We hope to have convincingly argued that direct renditions of flexible, constraint based parsing formalisms can be made to run efficiently while preserving a one to one correspondence between the conceptual and the representational levels, including for such non traditional formalisms as Property Grammars, in

which category inference does not depend on hierarchical or even constituency notions.

The representations allowed by our methodology, while extremely close to the computer-independent, conceptual representations of these formalisms, are directly executable, and moreover non-deterministic. This is satisfying with respect to logic programming's original aims of declarativeness and higher level expressiveness. Together with the advantages of this approach, we are able to even produce hierarchical depictions of the parse history of any category, including "incorrect" or incomplete ones. This is not important in itself, but is provided as an easy side effect, in the interest of historic comfort : we are all used to thinking in terms of parse trees or graphs, so showing a parse record in graph form may prove convenient to some users.

It is interesting to note that the parsing methodology we describe here has been generalised into a concept formation system which provides a cognitive sciences view of problem solving [Dahl04c].

Acknowledgements

This research was made possible by V. Dahl's NSERC Discovery and Equipment grants and P. Blache's CNRS-SdI grant. Thanks are due to Baohua Gu for his help with testing and debugging the parser.

References

[Abdennadher98] Abdennadher S. and Schütz H. (1998) "CHR: A flexible query language", in proceedings of *Int. Conference on Flexible Query Answering Systems*, volume 1495 of LNCS, Springer-Verlag.

[Barranco05] Barranco-Mendoza. A. (2005) *Stochastic and Heuristic Modelling for Analysis of the Growth of Pre-Invasive Lesions and for a Multidisciplinary Approach to Early Cancer Diagnosis*, PhD Dissertation, Simon Fraser University.

[Bès99a] Bès G & P. Blache (1999) "Propriétés et analyse d'un langage", in proceedings of *TALN'99*.

[Bès99b] Bès G., (1999) "La phrase verbale noyau en francais". In Recherches sur le francais parlé, GARS, 15, 273-358.

[Blache95] Blache P. & N. Hathout (1995) "Constraint Logic Programming for Natural Language Processing",in proceedings of *NLULP'95*.

[Blache01] Blache P. & J.-M. Balfourier (2001) "Property Grammars: a Flexible Constraint-Based Approach to Parsing", in proceedings of *IWPT-2001*.

[Blache05] Blache P. (2005), "Property Grammars: A Fully Constraint-Based Theory", in H. Christiansen & al. (eds), *Constraint Solving and NLP*, Lecture Notes in Computer Science, Springer.

[Christiansen01] Christiansen, H. (2001) "CHR as grammar formalism, a first report", Sixth Annual Workshop of the ERCIM Working Group on Constraints.

[Christiansen02a] Christiansen, H. (2002) *CHR Grammar web site*, http://www.ruc.dk/~henning/chrg

[Christiansen02b] Christiansen, H. (2002) "Logical Grammars Based on Constraint Handling Rules", in Proc. *18th International Conference on Logic Programming*, Stuckey, P. (ed.) Lecture Notes in Computer Science, 2401, Springer-Verlang, p. 481 .

[Christiansen02c] Christiansen, H. "Abductive Language Interpretation as Bottom-up Deduction", in Proc. *NLULP 2002, Natural Language Understanding and Logic Programming*, Wintner, S. (ed.), Copenhagen, Denmark, pp. 33–47.

[Christiansen05a] Christiansen, H. (2005) "CHR Grammars" in the *International Journal on Theory and Practice of Logic Programming*, special issue on Constraint Handling Rules, pp. 227-248.

[Christiansen05b] Christiansen, H. and Dahl, V. (2005) "HYPROLOG: a new logic programming language with assumptions and abduction", in Proceedings *ICLP'05 (International Conference on Logic Programming*, Sitges, Spain.

[Dahl97] Dahl, V., Tarau, P. and Li, R. (1997) "Assumption Grammars for Natural Language Processing". in *Lee Naish (ed.) Proc. Fourteenth International Conference on Logic Programming*, MIT Press, 1997.

[Dahl04a] Dahl V. "Treating Long-Distance Dependencies through Constraint Reasoning" (2004) in proceedings of *First International Workshop on Constraint Solving for Language Processing(CSLP'04)*.

[Dahl04b] Dahl, V. and Blache, P. (2004) "Directly Executable Constraint Based Grammars", in Proc. *Journees Francophones de Programmation en Logique avec Contraintes*, Angers, France, 149–166.

[Dahl04c] Dahl V. and Voll K. (2004) "Concept Formation Rules: an executable cognitive model of knowledge construction", in proceedings of *First International Workshop on Natural Language Understanding and Cognitive Sciences*, INSTICC Press.

[Dahl04d] Dahl, V. and Tarau, P. (2004) "Assumptive Logic Programming", in Proc. *ASAI 2004*, Cordoba, Argentina.

[Dalrymple91] Dalrymple, M., Shieber, S., and Pereira, F. (1991) "Ellipsis and Higher-Order Unification", In *Linguistics and Philosophy*, 14(4), 399–452

[Frühwirth98] Frühwirth T. (1998) "Theory and Practice of Constraint Handling Rules", in *Journal of Logic Programming*, 37:1-3.

[Maruyama90] Maruyama H. (1990), "Structural Disambiguation with Constraint Propagation", in proceedings of

[Mel'čuk88] Igor Mel'čuk (1988) "*Dependency Syntax*", SUNY Press.

[Morawietz00] Morawietz F. (2000) "Chart parsing as contraint propagation", in proceedings of *COLING-00*.

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

[Osborne99] M. Osborne (1999) "MDL-based DCG Induction for NP Identification", in proceedings ov *CoNLL-99*

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

[Shieber95] Shieber S., Y. Schabes & F. Pereira (1995) "Principles and implementation of deductive parsing", in *Journal of Logic Programming*, 24(1-2):3-36.

[Tesnière59] Tesnière L. (1959) *El/'ements de syntaxe structurale*, Klincksieck.

[Tjong00] Tjong Kim Sang E., W. Daelemans, H. Déjean, R. Koeling, Y. Krymolowski, V. Punyakanok & D. Roth (2000) "Applying System Combination to Base Noun Phrase Identification", in proceedings of *COLING-00*.