Constraints, Linguistic Theories and Natural Language Processing

Philippe Blache

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

1 Introduction

The notion of constraints is generally used in modern linguistics (in particular in syntax and phonology) for representing properties that an object must satisfy (see [Borsley96], [Sag99]). Constraints can be general (or universal), valid for different languages, or at the opposite very specific, representing for example the variability of a given language. In all cases, the idea consists of stipulating properties ruling out structures which don't belong to the language.

Most linguistic theories now integrate this notion, in particular constraint-based approaches (HPSG being the theory making the most intensive use of this notion), but also in the principle and parameters paradigm (in particular Optimality Theory, see[Archangeli97]). Even dependency grammars propose a constraint-based version called Constraint Dependency Grammars (see [Maruyama90]). However, the interpretation of this notion can be very different from one approach to another.

It is interesting to note that constraints are also used in computer science (see [Marriott98], [Saraswat93]), and logic programming in particular. The question addressed here concerns precisely the adequation of the notion of constraints in linguistics and in computer science (see [Guenthner88], [Meurers98], [Morawietz97]). We want to show that, while a superposition of both points of view is possible, the linguistic interpretation doesn't exploit all the properties of a constraint system. Concretely, several approaches in parsing try to interpret the parsing process as a constraint satisfaction problem. However, in most cases, constraints are used in a passive sense.

We show here that this problem comes in particular from the generative interpretation of the relation between grammar and language. More precisely, the derivation relation entails a conception of the parsing process consisting in building first a local structure (usually a tree) and then in verifying some properties over it. Then, the information allowing to build trees doesn't have the same status as the other linguistic knowledge. In such a perspective, constraints are relegated to a secondary role. We propose an approach representing all the information by means of constraints, at the same level and allowing to consider the parsing process as one of constraint satisfaction.

In a first section, we detail some examples illustrating different interpretations (and different uses) of constraints in linguistics. We then describe more precisely how constraints generally work within linguistic theories. Concretely several characteristics inhibit constraints from playing the same role as in computer science. The second section describes these limits and presents some properties that should be present in the definition of a constraint-based formalism. The third section proposes such a formalism, called *Property Grammars*, illustrating how constraints can form a system, useful both in a descriptive perspective and for parsing. The last section shows how such a system can be used for implementation.

2 Constraints in Linguistics

The following examples show different uses of constraints. They illustrate the variability of the representation level of constraints which can be used at a low level in order to verify properties or more generally, as part of the theory itself, the analysis mechanism becoming one of resolution.

2.1 *Constraint Grammars*

The notion of constraints as used in *Constraint Grammars* (see [Karlsson90]) is close to *if-then* rules. In this approach, constraints are applied to categories and express contextual dependencies. They are particularly useful to restrict the categorization mechanism.

1. (@w = 0 "REP" (-1 DET))
2. (@w = 0 VFIN (-1 TO))
3. ("that" =! "⟨ Rel ⟩" (-1 NOMHEAD) (1VFIN))

Example (1) indicates that if a word can be interpreted as a preposition and the preceding one (at position -1) is a determiner, then the "PREP" interpretation is ruled out. The same kind of constraint is represented in example (2) which stipulates that a word cannot be a finite verb before "TO". Such constraints express cooccurrence restrictions on the right and left contexts, as in the third example which indicates that "that" is a relative if it immediately follows a nominal head and precedes a finite verb.

One interesting aspect of these constraints is then the possibility of representing properties over objects with different granularities: objects can be specified as categories or subset of features (see [Samuelsson96]). This kind of technique is very efficient for POS tagging or text chunking. But representing linguistic information with such an approach can become difficult, in particular for the expression of generalizations.

2.2 *Constraint Dependency Grammar*

The use of constraints in dependency grammars was first proposed in [Maruyama90]. Dependency grammars build relations between words. CDG proposed some constraints over these relations.

1. $word(pos(x))=PP \Rightarrow$ ($word(mod(x)) \in \{PP, NP, V\}$, $mod(x) \prec pos(x)$)
2. $mod(x) \prec pos(y) \prec pos(x) \Rightarrow mod(x) \preceq mod(y) \preceq pos(x)$

In the first example, the constraint stipulates that if a word at position x is a PP, it can modify a PP, a NP or a V, these modified elements preceding the word in x. The second constraint implements projectivity and indicates that no crossing relations are allowed for modifications.

In this approach, constraints are not directly expressed over linguistic objects, but over relations. This entails a passive use of constraints in the sense that the parsing process consists in building first these relations and then verifying that they satisfy the constraints.

Some recent implementations of CDG control the satisfaction process by introducing a notion of weight over constraints (see [Menzel98], [Schröder00]) which, in a certain respect, comes to ranking constraints, like Optimality Theory. But this doesn't modify the fact that parsing is a two-stage process: determining relations, then satisfying constraints.

2.3 *HPSG*

In HPSG, principles, grammar rules and lexical entries are considered as constraints which interact in order to specify a syntactic structure. The theory relies explicitly on the notion of constraint satisfaction (cf. [Sag99]): "*A phrase structure is well formed just in case each local subtree within it either : (1) satisfies a lexical entry or (2) satisfies some grammar rule and all grammatical principles.*"

The following example presents a constraint on head-complement phrases. It stipulates that if a word takes complements (values of the COMPS feature), then the corresponding categories must appear as non-head daughters of the phrase headed by the word.

$$\left[\begin{array}{l} \textsc{synsem}\ \left[\textsc{syn}\ \left[\textsc{comps}\ \langle\rangle\right]\right] \\ \textsc{hd-dtr}\ \left[\begin{array}{l} word \\ \textsc{ss}\ \left[\textsc{syn}\ \left[\textsc{comps}\ \left\langle \boxed{1}, \ldots, \boxed{n}\right\rangle\right]\right] \end{array}\right] \\ \textsc{nhd-dtrs}\ \left\langle \left[\textsc{ss}\ \boxed{1}\right], \ldots, \left[\textsc{ss}\ \boxed{n}\right]\right\rangle \end{array}\right]$$

In such a view, there is no particular order for the verification of the information and the grammatical information is considered as a set of constraints. But, as explicit in the description of the grammaticality given above, this is also a two-stage mechanism which consists in first building a local tree (using tree schemas) and then verifying the other constraints.

3 Constraints in Linguistic Theories

As shown in the previous section, the use of constraints is of deep interest both from a descriptive and a formal point of view. However, in these examples, coming from three different formal paradigms, we can notice that constraints are passive in the sense that a structure has first to be built before verifying its properties. Practically, this has an impact on the scope of the constraint which can only be local and applied to a given structure. In other words, it is difficult to stipulate constraints over general variables, they have to be linked to a given structure. In constraint dependency grammars for example, relations have to be known before verifying constraints. The same is true in HPSG which requires a local structure (a local tree) before applying the principles.

In the case of constraint-based theories, this point comes from the generative interpretation of the relation between grammars and languages. Indeed, in this paradigm, a language is generated by a grammar and this relation is implemented by the derivation process. Even without explicit phrase-structure rules like in HPSG, this relation remains the core of the process: building a local tree (e.g. selecting a rule schema) is equivalent to deriving a non-terminal in classical PS grammars. This point is crucial because it is in a certain sense contradictory with an actual constraint-based view of the parsing process.

Now let us examine the computer science side. In constraint programming, the set of constraints forms a system which has to be verified at each step of the process (in constraint logic programming, for example, constraint resolution replaces unification). As soon as a variable needs to be instantiated, the coherence of the constraint system is verified before actually applying the instantiation. This means in particular that in constraint satisfaction problems, variables are accessible at each step (they have a global scope) in order to allow the coherence verification of the system and to control the evaluation all along the resolution: constraints in this sense are active and specify properties over the constrained objects during the process. Generally speaking, a constraint satisfaction process requires the possibility of expressing constraints over objets independently from the way to accessing them. Such an approach has interesting properties, in particular concerning declarativity: the description of the problem relies on the description of constraints, its resolution makes use of these constraints. Thus, describing a problem is equivalent to its resolution. This is one of the most important characteristics of constraint logic programming.

Concerning the particular problem of parsing, this means that constraints have to be specified directly over variables, independently from complex structures. In the examples described above, we encounter two different situations. Constraint grammars express constraints directly over words. However, it is difficult to figure out how these local contextual constraints can constitute a grammar. In the other example, constraints are not expressed directly over objects, but according to their situation within a structure (or a relation). In this case, we cannot consider that the set of constraints forms a system whose consistency can be checked globally: constraints form subsets (or subsystems) each being relevant to a substructure. The general process cannot then be considered as

a constraint satisfaction one[1]. In this sense, we can consider that constraints have a local use in the process. This limitation doesn't come from a particular restricted interpretation but from the theory itself.

Moreover, we can say that such a conception of constraints illustrates a particular conception of the role of grammars in linguistic analysis (this is particularly clear in natural language processing). The classical consideration consists in using grammatical information in order to separate the well-formed structures from the others. In this approach, constraints are imperative and used to implement the notion of grammaticality. However, it is important to notice that one of the arguments for using constraints in computer science is the possibility of building approximate solutions: the constraint system at the end of the process, even when it cannot be simplified to a unique solution, contains a lot of information. In other words, we can consider that the state of the constraint system constitutes in itself a solution, whatever its form. Concerning natural language processing, this means the possibility of associating information to an input, whatever its grammaticality status.

4 What an constraint-based approach should be

We have seen in the previous section that even if constraints play a crucial role in modern linguistic theories, we cannot totally conceive the parsing problem as one of constraint satisfaction. The main consequence is that a grammar cannot be viewed as a set of constraints containing all the information. As shown before, a parsing process in these approaches consists first in applying a derivation (i.e. building a local structure) and then verifying some constraints.

We present in the following a set of characteristics that should be present in a constraint-based approach.

Constrained objects: the objects affected by a constraint have to be accessible independently from structural information. In other words, it is necessary to express constraints over every kind of object (whatever its level) without needing the knowledge of a local structure for accessing its value. This condition is necessary to interpret the grammatical information as a constraint system and the parsing process as a constraint satisfaction problem. Otherwise, constraint resolution only constitutes a subpart of the entire mechanism, which is contradictory with the goals of a constraint-based theory.

In the same perspective, the constraints have to be expressed independently from the form of the objects and reciprocally, the structure of an object should not depend on the constraint that can affect it.

Constraint system: All the information has to be represented by means of constraints, for the same reasons as for the accessibility of values described above. A grammar must form a system of constraints which contains all the information

[1] It is then false to consider that Maruyama and CDG allows the interpretation of parsing as a constraint satisfaction problem.

required for a linguistic analysis. This is imperative if one want to interpret parsing as a CSP.

A constraint system is a set: All the constraints are represented and treated at the same level. There is no hierarchization nor sequencing of the constraints. The reason is that, again, the mechanism of constraint resolution must only rely on information contained by constraints[2] . With this property, a parsing mechanism can make use exclusively of constraints.

Encapsulation: Same information should not be disseminated through different representations. This means that the information represented in a constraint must be homogeneous and reciprocally that this information be linguistically motivated.

Grammaticality vs. characterization: As shown in the previous section, constraints are used for the verification of the well formedness of a local structure. Reciprocally, when no local structure can be built, constraints cannot be checked. The analysis problem is then restricted to the grammaticality question. It seems however interesting to take advantage of the notion of constraints in order to allow approximate solutions. In this case, the main goal of the analysis is the characterization of an input by means of linguistic properties rather than the verification of its grammaticality and the construction of an associated syntactic representation.

5 Property Grammars

We propose a formalism, called *Property Grammars*, implementing the advantages of using constraints. In this approach, all the information is represented by means of properties (see [Bès99]) which form a system of constraints. The properties are expressed over categories, which can be hierarchized. However, properties do not depend on a particular local structure. In this sense, the grammar forms an actual set of constraints. This also means that this approach is equational rather than generative: there is no derivation between the grammar and the described language.

In property grammars (as should be the case for actual constraint-based theories), the notion of grammaticality is not crucial. More precisely, grammaticality is replaced with a more general notion: *characterization*. Concretely, the characterization of an input corresponds to the state of the constraint system at the end of the parse. We will see in the following that such a result can be interpreted as a unique structure (which can for example be represented as 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.

[2] Constraint ranking as used in OT contains in fact implicit information. Moreover, it can be reduced in certain cases to a simple default mechanism more than an actual parsing process.

5.1 Properties

In this approach, properties (which are synonyms to constraints) express relations between categories. A category is, classically in a constraint-based approach, a set of hierarchized features. The propagation of feature values from one category to another is explicit in the constraints.

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 *Property Grammars*. As said before, a property is expressed independently from any local structure, but directly between categories. Obviously, for clarity, it is interesting to 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 (together with their notations). 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[3].

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

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

  The corresponding categories (in which *Sup* stands for superlative) can be dominated in the hierarchy by a *NP*.
- **Obligation** (*oblig*): Specifies the set of compulsory, unique categories. Such categories correspond to the heads.

  Example: $oblig(VP) = \{V\}$
- **Unicity** (*uniq*): 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: $\{le\ être_{[N,P]}\} \overset{VP}{\Rightarrow} ClR_{[N,P]}$ (Je me le suis dit/*I told that to myself*)
  If the clitic *le* and a finite verb *être* (with agreement features N and P) cooccur (characterizing a VP), then a reflexive clitic (which agrees with the verb) is needed.
- **Exclusion** ($\not\Rightarrow$): Restriction of cooccurrence between sets of categories.

  Example: $Clit[Ref] \overset{VP}{\not\Rightarrow} lui$ (*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.

  Example: $Det \prec AP$ $\qquad$ $AP \prec N$

[3] Examples are in French.

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

Example: $Det \rightsquigarrow N$ $\quad$ $AP \rightsquigarrow N$

These properties contain the basic syntactic information. Other properties can be added if necessary, in particular for the integration of 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 the objects values: we need to know the set of categories associated to an input. We have shown that building a local structure before verifying constraints has negative consequences on the use of constraints. However, 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[4]. 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 the hierarchical information between a particular category and an upper-level indicated by such a property is not used in the satisfaction process, as we will see it in the next section.

The satisfiability process 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.

5.2 An example

We give in this section an example of description of the *NP* in French (including the superlative). This grammar (see figure (1)) illustrates the use of the different properties. As mentioned in the previous section, the properties are grouped

[4] The knowledge of the entire set of categories actually depends on the syntactic structuration. Preferring flat structures allows an immediate access to this set of categories.

according to the syntactic unit they describe, but this is merely for readability. All constraints are at the same level, not hierarchized and the entire grammar has to be considered as a constraint system in which constraints can be verified independently from each other. In the same order of idea, the indexation doesn't play a particular role, but is used in the following to refer to the constraints.

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) $Det \prec Sup$
(8) $N \prec Sup$
(9) $AP \not\Rightarrow Sup$
(10) $Det \rightsquigarrow N$
(11) $AP \rightsquigarrow N$
(12) $Sup \rightsquigarrow N$

Properties of the AP

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

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) $Det \rightsquigarrow Adj$
(9) $Adv \rightsquigarrow Adj$

Fig. 1. A property grammar for the NP

Let's give some precisions on this grammar. Constraint (3) of the NP stipulates that a common noun requires a determiner. This is the classical subcategorization information for which syntactic aspects are represented separately from semantic ones (using exigency and dependency constraints). Concerning the dependency relation, we can also notice that all kinds of category (not necessarily the head) can be the target of the relation. In constraint (2), we can see that an AP can be the head of the NP which can be the case in examples such as *"donne-moi la rouge (give me the red)"*. In this case, we would need another dependency constraint (not represented here) indicating that Det depends on the AP. Constraint (9) stipulates that an AP cannot cooccur with a superlative as shown in the examples:

(1) a. le vieux livre. (*the old book*)
    b. le livre le plus vieux. (*the oldest book*)
    c. * le vieux livre le plus cher.

Finally, concerning the superlative, the property (4) indicates that an adverb necessarily belongs to this unit without being a possible head.

5.3 Parsing

We describe in this section the general parsing mechanism. Let's take an example relying on the grammar of the NP described in the previous section:

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

Step 1 (Categorization): The first stage consists in finding the categories associated with the words of the input. We don't need to disambiguate this operation, the goal being to build all the possible characterizations of the input. Each category is associated with the position of the corresponding word, we obtain a set of pairs of the form $\langle cat, pos \rangle$. For clarity, we note these pairs as cat_{pos}. For our example, we build :

$\Sigma_1 = \{ Det_1, Pro_1, N_2, V_2, Det_3, Pro_3, Adv_4, Adj_5 \}$

Step 2 (Determining all the objects): The second operation consists in building the set of all the relevant categories for the input. This amounts to defining a set of variables in which the constraints will be applied. This operation consists in finding for each category the *constituency* property to which it belongs. These properties indicate the dominating syntactic unit which will then belong to the set of possible objects. We build this new set from the first set Σ_1:

$\Sigma_2 = \{ Det_1, Pro_1, NP_1, Sup_1, N_2, NP_2, V_2, Det_3, Pro_3, NP_3, Sup_3, Adv_4, AP_4, Sup_4, Adj_5, AP_5, Sup_5, \}$

Step 3 (Building characterizations): Characterizing an input consists in building the set of all subsets of Σ_2 and verifying constraints for each subset. Concretely, it is possible to associate to these subsets the state of the relevant constraints described by P^+ (the set of satisfied constraints) and P^- (the set of unsatisfied constraints). Figure (2) presents some characterization sets (i.e. characterizations associated to some subsets of Σ_2).

- Det_1/N_2 :	$P^+(NP) = \{1, 2, 3, 4, 10\}$	$P^-(NP) = \emptyset$
- Sup_1/N_2 :	$P^+(NP) = \{1, 2, 12\}$	$P^-(NP) = \{6,3\}$
- Det_3/Sup_4 :	$P^+(NP) = \{1, 6\}$	$P^-(NP) = \{2\}$
- Det_3/AP_4 :	$P^+(NP) = \{1, 5\}$	$P^-(NP) = \{2\}$
- Sup_3/AP_4 :	$P^+(NP) = \{1\}$	$P^-(NP) = \{2, 9\}$
- Adv_4/Adj_5 :	$P^+(AP) = \{1, 2, 3, 4\}$	$P^-(AP) = \emptyset$
	$P^+(Sup) = \{1, 2, 4, 9\}$	$P^-(Sup) = \{3\}$
- $Det_1/N_2/Sup_3$:	$P^+(NP) = \{1\text{-}4, 6, 8, 9, 10, 12\}$	$P^-(NP) = \emptyset$
- $Sup_1/N_2/Det_3$:	$P^+(NP) = \{1, 2, 3, 9, 10, 12\}$	$P^-(NP) = \{4, 6, 8\}$
- $Det_3/Adv_4/Adj_5$:	$P^+(Sup) = \{1, 2, 3, 4, 5, 6, 8, 9\}$	$P^-(NP) = \emptyset$
- $Det_3/AP_4/Sup_5$:	$P^+(Sup) = \{1, 4, 5, 6\}$	$P^-(NP) = \{2, 9\}$
- $Det_1/N_2/Sup_3/AP_4$:	$P^+(NP) = \{1\text{-}8, 10, 11, 12\}$	$P^-(NP) = \{9\}$

Fig. 2. Characterization sets

We can remark that all subsets can be characterized, some of them having an empty set P^-. In all cases, whatever the form of the input, a precise description can be given in terms of satisfied and unsatisfied properties. Some subsets can

have different characterizations: the subset Adv_4/Adj_5 can be characterized using relevant properties of AP and Sup.

If characterizations are resticted to that receiving an empty P^-, then only grammatical constructions are analyzed. In our example, the positively characterized subsets are: $\{Det_1/N_2,\ Adv_4/Adj_5,\ Det_1/N_2/Sup_3,\ Det_3/Adv_4/Adj_5\}$. Finding a sequence covering the total input leads to a syntactic structure formed by the last two subsets.

6 Conclusion

Property grammars rely entirely on constraints, both from the linguistic and the computational point of view: all the syntactic information is represented by means of properties, the parsing process being one of constraint satisfaction. The main characteristics of this approach is that constraints can be expressed generally, independently from the form of the syntactic representation. In other words, it is not necessary in this approach to build a local structure and then verify constraints. All the objects being available before the parsing process itself, a grammar can play the role of an actual constraint system and linguistic properties can be verified independently.

One of the main interests of this approach is the possibility of using the general notion of *characterization* in place of that of *grammaticality*. It is then possible to analyze any kind of input, well-formed or not. This is of great importance for real-life natural language processing applications such as dialogue systems. In fact, such an approach proposes a more realistic view for the sudy of language in which a syntactic structure is not necessarily a homogeneous structure but can be constituted by several (possibly disconnected) pieces of information.

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 actes de TALN'99.

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

[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*.

[Götz97] Götz T., D. Meurers & D. Gerdemann (1997), *The ConTroll Manual*, SFS Report.

[Guenthner88] Guenthner F. (1988) "Features and Values 1988", *CIS-Bericht-90-2*, Universität München.

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

[Marriott98] Marriott, K. & Stuckey, P. J. (1998) *Programming with Constraints*, MIT Press.

[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.

[Morawietz97] Morawietz F. & Cornell T. (1997) "Representing Constraints with Automata", in proceedings of the ACL/EACL.

[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*.

[Saraswat93] Saraswat V. (1993), *Concurrent Constraint Programming*, MIT Press.

[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.