Constraints and Language

Constraints and Language

Edited by

Philippe Blache, Henning Christiansen,
Verónica Dahl, Denys Duchier
and Jørgen Villadsen

CAMBRIDGE
SCHOLARS
PUBLISHING

Constraints and Language,
Edited by Philippe Blache, Henning Christiansen, Verónica Dahl,
Denys Duchier and Jørgen Villadsen

This book first published 2014

Cambridge Scholars Publishing

12 Back Chapman Street, Newcastle upon Tyne, NE6 2XX, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-4438-6052-2, ISBN (13): 978-1-4438-6052-9

CONTENTS

List of Illustrations **xi**

List of Tables **xiii**

Preface **xv**

I Foundations and Overview **1**

1 Constraints in (Computational) Linguistics
Philippe Blache, Jørgen Villadsen **3**
1.1 Introduction 3
1.2 Constraints and programming 4
1.3 Constraints, linguistics and parsing 6
1.3.1 Constraints on trees: active constraints and parsing . 8
1.3.2 GPSG: the separation of information 10
1.3.3 HPSG: the notion of satisfaction 11
1.3.4 OT: relaxing constraints 13
1.3.5 PG: constraints as syntactic structure 15
1.4 Conclusion 17
Bibliography 18

2 Constraints and Logic Programming in Grammars and Language Analysis
Henning Christiansen **21**
2.1 Introduction 21

2.2 Background 22
2.2.1 Different Notions of Constraints in Grammars and Logic Programming 22
2.2.2 Constraint Handling Rules, CHR 23
2.3 Abductive Reasoning in Logic Programming with Constraints 25
2.4 Using CHR with Definite Clause Grammars for Discourse Analysis 28
2.5 CHR Grammars 30
2.6 Conclusion 33
Bibliography 33

3 Model-theoretic Syntax: Property Grammars, Status and Directions
Philippe Blache, Jean-Philippe Prost **37**
3.1 Introduction 37
3.2 Model Theory for Modelling Natural Language 38
3.3 The Constructive Perspective: A Constraint Network for Representing and Processing the Linguistic Structure 40
3.3.1 Generative-Enumerative vs. Model-Theoretic Syntax 41
3.3.2 Generativity and hierarchical structures 43
3.3.3 The Property Grammar Framework 46
3.4 The Descriptive Perspective: A Constraint Network for Completing the Linguistic Structure 49
3.5 Grammaticality Judgement 52
3.6 Conclusion 55
Bibliography 56

4 Constraints in Optimality Theory: Personal Pronouns and Pointing
Helen de Hoop **61**
4.1 Introduction 61
4.2 First and second versus third person pronouns 64
4.3 Incremental optimisation of anaphoric third person pronouns 70
4.4 OT semantic analysis of personal pronouns and pointing . . 72
4.5 OT syntactic analysis of personal pronouns and pointing . . 81
4.6 Conclusion 84
Bibliography 85

II Recent Advances in Constraints and Language Processing 91

5 Constraint-driven Grammar Description
Benoît Crabbé, Denys Duchier, Yannick Parmentier, Simon Petitjean **93**
5.1 Introduction . . . 93
5.2 Semi-automatic production of tree-adjoining grammars . . . 97
5.2.1 Lexical rules . . . 97
5.2.2 Description languages . . . 98
5.3 eXtensible MetaGrammar: constraint-based grammar description . . . 100
5.3.1 A language for describing tree fragments . . . 100
5.3.2 A language for combining tree fragments . . . 106
5.3.3 Towards a library of linguistic principles . . . 107
5.4 Cross framework grammar design using metagrammars . . . 109
5.4.1 Producing a lexical-functional grammar using a metagrammar . . . 111
5.4.2 Producing a property grammar using a metagrammar 114
5.4.3 Towards extensible metagrammars . . . 117
5.5 Conclusion . . . 118
Bibliography . . . 118

6 Extending the Constraint Satisfaction for better Language Processing
Kilian A. Foth, Patrick McCrae, Wolfgang Menzel **123**
6.1 Introduction . . . 123
6.2 The Constraint Satisfaction Problem . . . 126
6.3 NLP formalisms and the CSP . . . 128
6.3.1 Constraints as value subsets . . . 128
6.3.2 Hard and soft Constraints . . . 130
6.3.3 Uniform and free-form Constraints . . . 131
6.3.4 Axiomatic and empirical grammars . . . 132
6.4 Dependency Grammar Modelling with Locally-Scoped Constraints . . . 133
6.5 Expressivity of WCDG . . . 134
6.6 Extending Local Constraints to Global Phenomena . . . 135
6.6.1 Supra-local Constraints . . . 135
6.6.2 Recursive Tree Traversal . . . 138
6.6.3 Localised Ancillary Constraints . . . 139
6.6.4 Cascading and Recursive Ancillary Constraints . . . 141

6.7 Conclusions 145
6.8 Future work 146
Bibliography 146

7 On Semantic Properties in Constraint-Based Grammars
Verónica Dahl, Baohua Gu, J. Emilio Miralles **149**
7.1 Introduction 149
7.2 Background on Property Grammars 151
7.3 Semantic Property Grammars 152
7.4 Our Parsing Methodology 154
7.4.1 Background: HyProlog 154
7.4.2 A Hyprolog Parser for Property Grammars 157
7.5 Related Work 159
7.6 Conclusion 160
Bibliography 161
Appendix 164

8 Multi-dimensional Type Theory: Rules, Categories and Combinators for Syntax and Semantics
Jørgen Villadsen **167**
8.1 Introduction 167
8.1.1 Background 168
8.1.2 Arguments 168
8.1.3 Formulas 169
8.1.4 Strings 169
8.1.5 Combinators 170
8.1.6 Type Language and Type Interpretation 171
8.1.7 Theory of Inhabitation and Theory of Formation . . 172
8.1.8 Nabla 173
8.2 The Rules 173
8.2.1 Comments 175
8.3 The Categories 176
8.3.1 Comments 177
8.4 The Combinators 177
8.4.1 Comments 178
8.5 Examples: Syntax and Semantics 180
8.5.1 Step-by-Step Formula Extraction 181
8.5.2 Further Examples 183
8.6 Conclusion 185
Bibliography 186

9 Constraint-based Sign Language Processing
Annelies Braffort, Michael Filhol **191**
9.1 Introduction . . . 191
9.2 Linguistic description of Sign Languages . . . 192
9.2.1 Phonology . . . 193
9.2.2 Phonetics . . . 193
9.2.3 Lexicon . . . 194
9.2.4 Lexicon, syntax... and linguistic levels . . . 196
9.3 Language models . . . 197
9.3.1 Generative/categorical grammars . . . 197
9.3.2 Machine learning approaches . . . 198
9.3.3 SL-specific approaches . . . 199
9.4 AZee . . . 204
9.5 KAZOO . . . 210
9.5.1 SL Generation Module (SL Gene) . . . 211
9.5.2 Virtual Signer Animation Module (VS Anim) . . . 212
9.6 Conclusion . . . 214
Bibliography . . . 214

10 Geometric Logics
Hedda R. Schmidtke **219**
10.1 Introduction . . . 220
10.2 Geometric Semantics . . . 221
10.3 Expressiveness of Context Logic . . . 226
10.3.1 Binary Relations in Context Logic . . . 227
10.3.2 Perception and Reasoning . . . 228
10.3.3 Changing Perspectives . . . 229
10.4 Conclusions . . . 231
Bibliography . . . 232

III Applications **235**

11 Constraint-based Word Segmentation for Chinese
Henning Christiansen, Bo Li **237**
11.1 Introduction . . . 237
11.2 Background and Related Work . . . 238
11.2.1 The Chinese Word Segmentation Problem . . . 238
11.2.2 CHR Grammars . . . 239
11.3 A Lexicon in a CHR Grammar . . . 242

11.4 Maximum Matching 243
11.5 Maximum Ambiguous Segments 245
11.6 Discussion 247
11.7 Conclusion 248
Bibliography 249

12 Supertagging with Constraints
Guillaume Bonfante, Bruno Guillaume, Mathieu Morey, Guy Perrier **253**
12.1 Introduction 253
12.2 The Companionship Principle in Brief 257
12.2.1 Parsing with an AB-grammar 257
12.2.2 Filtering lexical taggings with the Companionship Principle 258
12.2.3 Implementation with Automata 261
12.3 Lexicalised Grammars 261
12.4 The Companionship Principle 268
12.4.1 The statement of the Companionship Principle . . . 268
12.4.2 The "Companionship Principle" language 269
12.4.3 Generalisation of the Companionship Principle to abstraction 270
12.4.4 The Undirected Companionship Principle 272
12.4.5 The Affine and Linear Companionship Principles . . 273
12.5 Implementation of the Companionship Principle with automata 276
12.5.1 Automaton to represent sets of lexical taggings . . . 276
12.5.2 Implementation of the Companionship Principle . . 277
12.5.3 Approximation: the Rough Companionship Principle (RCP) 280
12.5.4 Affine and Linear Companionship Principle (ACP) . 281
12.5.5 Implementation of Lexicalised grammars 283
12.6 Application to Interaction Grammars 284
12.6.1 Interaction Grammars 284
12.6.2 Companionship Principle for IG 287
12.7 Application to Lexicalised Tree Adjoining Grammars (LTAG) 290
12.8 Conclusion 293
Bibliography 294

Contributors **299**

Index **305**

List of Illustrations

1.1 Example of a feature-structure satisfying an input description 13

3.1 Example of parse deemed ungrammatical through model checking 51
3.2 Tree model for a NP constituent 53
3.3 Tree model for a S constituent 54

4.1 Pointing to two imaginary addressees $[\text{you}]_1$ and $[\text{you}]_2$ (Zwets 2014) 73
4.2 Pointing to introduce a discourse referent (Zwets 2014) . . . 74
4.3 Pointing at drawing while uttering sentence (4.4.3) 76
4.4 Pointing to an imaginary addressee while uttering sentence (4.4.4) (Zwets 2014) 78
4.5 Pointing at paper while uttering a sentence (4.4.5) (Zwets 2014) 79

5.1 Tree rewriting in TAG 95
5.2 TAG tree templates 96
5.3 Structural redundancy in TAG 96
5.4 Combining elementary tree fragments 99
5.5 Fragments described using the language $\mathrm{L}(\emptyset)$ 102
5.6 Models for the combination of fragments of Figure 5.5 . . . 102
5.7 Fragments described using the language $\mathrm{L}(\emptyset)$ (continued) . . 103
5.8 Models for the combination of fragments of Figure 5.7 . . . 103
5.9 Fragments described using the language $\mathrm{L}({}_{g}names)$ 104

5.10 Description of double PP complementation 104
5.11 Combination scheme for $L(colours)$ 105
5.12 Fragments described using the language $L(colours)$ 106
5.13 LFG grammar and c-and f-structures for the sentence "John loves Mary" 112
5.14 Fragment of a PG for French (basic verbal constructions) . . 115

6.1 AGENT assignment in German perfect tense active sentences 141
6.2 AGENT assignment in German passives 143
6.3 AGENT assignment based on German active/passive detection 145

7.1 New Category Inference 164
7.2 A sample ontology of biological concepts that have IS-A relation. 166

9.1 The sign BALL in its citation form (a) (source: Dictionnaire bilingue LSF, Editions IVT), and the parameters that can vary depending on the context (b). 195
9.2 The sign CAR PARK (source: Dictionnaire bilingue LSF, Editions IVT) 196
9.3 The four manual units composing S_1 translated in LSF . . . 201
9.4 Signed example sentence S_1 modelled with P/C and null nodes 201
9.5 Rule time line illustrations (parameter arguments are in italics) 204
9.6 The sign HELLO, THANK YOU 205
9.7 An AZee output of type SCORE 206
9.8 Kazoo: module organisation. 211
9.9 Kazoo demo page, version 1.0. 213

12.1 The LTA of sentence (1) before and after filtering with the Companionship Principle 262
12.2 Automaton $\mathscr{A}$ for the last constraint of Table 12.4 (page 261) 279
12.3 The PT of Sentence (2) 285
12.4 PTDs for Sentence (2) 286
12.5 Polarity composition 286
12.6 Filtering in Interaction Grammars 289
12.7 Filtering in Tree Adjoining Grammars 293

List of Tables

4.1 Incremental optimisation of the interpretation of *he*, stage 1, sentence (4.3.1) . . . 71
4.2 Incremental optimisation of the interpretation of *he*, stage 2, sentence (4.3.1) . . . 72
4.3 Interpretive optimisation of pointing in sentence (4.4.3), Figure 4.3 . . . 77
4.4 Interpretive optimisation of pointing in sentence (4.4.4), Figure 4.4 . . . 79
4.5 Interpretive optimisation of pointing in sentence (4.4.5), Figure 4.5 . . . 80
4.6 Interpretive optimisation of pointing in sentence (4.4.2), Figure 4.2 (page 74) . . . 80
4.7 Expressive optimisation of reference to the speaker in sign language context . . . 82
4.8 Expressive optimisation of reference to the speaker in spoken language context . . . 83
4.9 Expressive optimisation of reference to multiple addressees in spoken language context . . . 83

6.1 Operators in WCDG . . . 134

12.1 Toy lexicon of an AB-grammar . . . 258
12.2 Types of the grammar redefined with a flat structure . . . 260
12.3 Head companions . . . 260
12.4 Argument companions . . . 261

PREFACE

This book is addressed to scholars and students of linguistics and computational linguistics as well as others. Constraints are fundamental notions in the characterisation and processing of language. A model of language may consist of a generative and a constraining part, independently of whether the model concentrates on syntax or on the relationship between the meanings and their expression, and some models are entirely based on constraints. Different sorts of constraints appear in studies of languages, in computational linguistics and in a variety of programming paradigms. This book does not claim to provide a unified view of constraints, but aims at creating a mutual inspiration and transfer of results between the different fields and directions covered in this book.

Constraint programming emerged due to a need to solve complex, mathematically formulated problems, including optimisation problems, and – especially in its variants of constraint logic programming – has provided an additional expressibility that is very useful for applications on language. Generative grammar formalisms, typically with a context-free backbone, have simple grammatical rules or complex attributes to capture semantic properties, and constraints can express concordance and formalise flow of information between different subphrases. Completely constraint based systems include Optimality Theory and Property Grammars that are also described in this book.

Language is considered from a general perspective, ranging from human languages such as written, spoken or signed languages, over biological sequence data to streams of sensor signals received by a robot or an ambient intelligent computer application. They are all systems of encoded meanings

in some syntactic form and present analogous problems of characterising them as well as concretely extracting meanings from expressions.

This book arises from the series of workshops on *Constraint Solving and Language Processing* which started in 2004 and have been held with varying intervals since then. Proceedings have been published in Lecture Notes in Computer Science in 2004, and from 2012 on they appear in the FoLLI sub-series of Lecture Notes in Computer Science. Proceedings from the intermediate years have been issued as technical reports that are available online; a complete list is maintained at `http://www.ruc.dk/~henning/CSLP_AllWorkshops/`.

The first workshop took place at Roskilde University, Denmark, as part of a research project funded by the Danish Natural Science Research Council, lead by Henning Christiansen with the participation of Philippe Blache, Verónica Dahl, Jørgen Villadsen and the late Peter Rossen Skadhauge. Since then, these workshops have taken place in different cities of the world, Sitges in Spain, Sydney in Australia, Hamburg in Germany (with the ESSLLI summer school); after a pause of a few years, the series was resumed in 2012 due to an initiative by Denys Duchier and Yannick Parmentier in Orléans, France. The editors would like to thank the participants, program committee members, organisers and the long row of prominent invited speakers of these workshops, and especially those who contributed to this book. Finally, we are grateful to the Viking Ship Museum, Roskilde, Denmark, for allowing us to use the photo of its Viking longship copy, the Sea Stallion of Glendalough, for the book cover.

The chapters of this book are divided into three parts. Part I provides foundations and overview of fields that are central to Constraints and Language, parts II and III presents a collection of recent results and applications.

HENNING CHRISTIANSEN
Roskilde, July 2014

Part I

Foundations and Overview

CHAPTER ONE

CONSTRAINTS IN (COMPUTATIONAL) LINGUISTICS

PHILIPPE BLACHE, JØRGEN VILLADSEN

1.1 INTRODUCTION

The notion of constraints started to occupy a central position in linguistic theories from the introduction of unification in grammars. This evolution has been first done implicitly with so-called logic grammars, closely related with the history of Prolog: see for example Metamorphosis Grammars (Colmerauer, 1975); Definite Clause Grammars (Pereira and Warren, 1980) and Functional Unification Grammars (Kay, 1984). What is important with unification is that it has been a major step towards the introduction of constraints both in programming and in linguistics. Basically, unification can implemented with an equation system, as it is the case in Prolog II (Colmerauer, 1986). Variables being possibly any kind of objects, unification becomes on the one hand a powerful mechanism to reduce the search space and on the other hand a way to represent high level relations between the objects or their characteristics. The introduction of unification in grammars arrived with the representation of linguistic items' properties by means of feature structures (see (Carpenter, 1992) for a precise description), enabling the implementation of relations not only between categories, but also between features. It became possible for example to represent directly different mechanisms such as sub-categorisation, agreement or lexical

selection. GPSG (Gazdar *et al.*, 1985) was among the first linguistic theories making an intensive use of feature structures and integrating the above mentioned processes. This theory constituted a major rupture with the dominant generative paradigm precisely for this reason: syntactic relations were not anymore represented only in terms of rules, but also by means of other statements. The first major innovation in GPSG is the separate representation of linear precedence. The second is the possibility to express directly feature co-occurrence restriction. These two mechanisms (among others) act as constraints on the structure introducing this idea that building a syntactic structure is not only a matter of derivation (in other words does not only relies on rules), but also on other kinds of relations between the linguistic objects. The introduction of constraints into linguistic theories became then obvious.

We propose in this chapter to explore the evolution of the notion of constraints in syntactic theories and how the computational and the linguistic perspective progressively get closer. In a first part, we propose an overview of constraint programming, showing how constraints not only introduce a way to control the processing (typically by reducing the search space), but also constitute an alternative way of representing and processing information (renewing the notion of declarativity). In a second part, we will detail the evolution of linguistic theories, from unification to constraints, showing how the notion of satisfaction can become the core of the theory. We will illustrate this evolution with the description of different theories, among which HPSG (Pollard and Sag, 1994), Optimality Theory (Prince and Smolensky, 1993) and Property Grammars (Blache, 2000). In the last section, we will situate this evolution in the model-theoretic perspective, and discuss how constraints can deeply renew our view of linguistic theory.

1.2 CONSTRAINTS AND PROGRAMMING

One way to present the notion of constraint relies on the state of the search space generated by a program. As classically presented (see e.g. (Saraswat and Rinard, 1990)), the state of a system is described by a *store*, which is the set of variables used in the program and a valuation function assigning each variable a value. A constraint provides partial information on the possible values. It specifies a subset of values in the initial domain, reducing then the state space. The conjunction of two constraints is the intersection of the set of values defined by each constraint. A typical constraint program consists in refining at each step the state of the search space

in a monotonic way: the set of possible values at one step is a subset of the possible values of the same variables at the prior step. A solution in a Constraint Satisfaction Problem (CSP, see (Jaffar and Lassez, 1986)) is an assignment of values so that all constraints are satisfied simultaneously.

An example over finite domains illustrates this process. Let's note I_1 an integer variable, and S_1 a set variable. The following example shows constraints over the two kinds of variables, where :

$$\boxed{\begin{array}{c} I_1 \in \{1,5\} \\ \{4,6,8\} \subseteq S_1 \end{array}} \tag{1.1}$$

Both constraints illustrate how the domain of the different variables can be reduced, implementing the representation of partial information about them. Describing a problem consists in stipulating the different constraints over the variables of interest into a *constraint store*. No value can be assigned to a variable without satisfying the constraint store. At each step of the process, the constraint store can be enriched with new constraints. Moreover, constraints interact as explained before: a same variable can be constrained with different stipulations, either directly or not. Constraint propagation consists then in evaluating the intersection of the domains specified by the different constraints, as illustrated in the following example:

$$\boxed{\begin{array}{c} \{4,6\} \subseteq S_1 \\ S_2 \subseteq \{1,2,4\} \\ S_3 \subseteq \{4,6,8\} \\ S_2 \subseteq S_3 \\ S_1 \subseteq S_3 \end{array}} \longrightarrow \boxed{\begin{array}{c} \{4,6\} \subseteq S_1 \subseteq \{4,6,8\} \\ S_2 \subseteq \{4\} \\ S_3 \subseteq \{4,6,8\} \end{array}} \tag{1.2}$$

In this example, constraint propagation makes it possible to reduce drastically the definition domain of the different variables. In some cases, we can see that this process can lead to a unique variable assignment, which is a solution satisfying the constraint system. This is one of the major interests of constraints: their interaction specifies *a priori* a reduced definition domain for each variable, which simply means that no value can be chosen outside it. This characteristic illustrates how constraints can be *active*: they are applied *a priori*, before doing any processing (Van Hentenryck, 1989). This is another major interest in constraint programming: the classical strategy in imperative programming consists in enumerating the possible values to assign before verifying their properties. This process correspond to the *generate-and-test* strategy. In constraint programming, active constraints as presented above make it possible to apply property verification before gen-

erating a value. Many other kinds of constraints can be stipulated, such as ordering, equality, arithmetic, etc.

To summarise, constraints are especially useful in two respects: information representation and control over the processes. In particular, they propose a direct way to represent partial information. Moreover constraint satisfaction being monotonic, all the different constraints are at the same level in the sense that they can be evaluated independently. Finally, constraint propagation provides an efficient way to control the processes by reducing the search space *a priori*. These different characteristics, plus the fact that there is a large variety of constraint types, make this approach well adapted to language processing.

1.3 CONSTRAINTS, LINGUISTICS AND PARSING

As said above, unification in linguistics opened the door to the introduction of constraints both for representing information (most linguistic theories now use this notion), but also in terms of computing: logic programming and the implementation of unification by means of an equation system constituted an adequate paradigm in which unification was considered as an active constraint. Both from theoretical and computational reasons, unification progressively gave the floor to constraints. This shift from unification to constraints in linguistics has been detailed by Pollard (1996). In this presentation, Carl Pollard identified the main properties shared by Constraint-Based Grammars (hereafter CBG), founding a new theoretical paradigm. We highlight in the following some of them:

- **Expressivity**: "*The language in which the grammatical theory is expressed does not impose constraints on the theory; it is the theory that imposes the constraints*."

  This property is directly related to what is called declarativity in programming: statements in the theory should not describe mechanisms on how to build the structure, but have to be linguistically motivated. For example, the description of how feature values are propagated or the kind of trees that are considered as valid should not belong to the theory. Theory and formalisms should be clearly distinguished.

- **Empirical Adequacy**: "*First write constraints that get the facts right, and worry later about which constraints are axioms and which are theorems. There are no deep principles.*"

In many cases in linguistics, facts require some axioms or principles that cannot be necessarily deduced or proved from anything else. What a linguistic theory should do is to describe as many facts as possible, even though axioms instead of theorems are required.

- **Locality**: "*Constraints are local in the sense that whether or not they are satisfied by a candidate structure is determined solely by that structure, without reference to other structures.*"

  This property means that constraints have to be evaluated independently from other considerations. In other words, constraints need to be stipulated for themselves, without being part of an operational architecture.

- **Psycholinguistic Responsibility**: "*Linguistic theories must be capable of interfacing with plausible cognitive models.*"

  In many cases, linguistic theories have been elaborated independently from the object they study - language- and its use by human subjects. Such theories became abstract formal objects. We will see that constraints can play an important role in the elaboration of cognitively grounded theories in the sense that they can represent linguistically motivated operations and they can be evaluated independently. In other words, their role can be observed in human language processing.

- **Radical Non-autonomy**: "*The grammar consists of assertions that mutually constrain several different domains. Some of these constraints may apply only to one domain. But typically, constraints are interface constraints.*"

  In this interpretation, the grammar is a set of constraints, whatever the domain (morphology, syntax, phonology, prosody, etc.). All domains are independent in the sense that none of them is the result of the transformation of another. Parsing results then from constraint interaction.

Another important view bringing a broader theoretical perspective to constraints in linguistics is the distinction made by Geoffrey Pullum between *Generative-Enumerative Syntax* (hereafter GES) and *Model-Theoretic Syntax* (hereafter MTS) (Pullum and Scholz, 2001); (Pullum, 2007)[1]. In this work, Pullum underlines some of the main properties of generative approaches that make them problematic when adopting a broad perspective

[1] See chapter 3 in this volume for a more precise presentation of MTS.

such as the one proposed by Pollard. One of these properties is the fact that GES relies on the idea that a grammar is a recursive definition of a set, then computably enumerable. Language processing relies there on a finite set of primitive elements and a finite set of operations for composing them into larger complex units.

This conception entails a specific relation between language and grammar in which the language is the set of derived strings generated from the grammar by means of derivation. In this perspective, language is recursively enumerable, which as a side effect means that we cannot say anything about elements that does not belong to the set. In GES, parsing consists then in finding a derivation which makes it possible to build a tree. This approach is holistic in the sense that the entire system is required for finding a derivation: no GES rule can be evaluated in itself, but is just a step in the derivation process.

On the opposite, a grammar in MTS does not recursively define a set of expressions: it merely states necessary conditions on the syntactic structure of expressions. The goal in MTS is to find an interpretation of grammatical information (not building a structure) or, in other words, to describe the characteristics of an input (not associating it with a structure). A MTS grammar is a set of independent assertions (formulas). Grammatical statements are here constraints on categories and a model is a set of categories satisfying the grammatical constraints.

Pollard's and Pullum's papers propose a new perspective for linguistic theories in which description of linguistic facts occupies a central position in the architecture. Instead of describing how to build a structure compatible with the observations, the question is to describe the facts thanks to grammatical statements that are satisfied by their description. In this theoretical framework, constraints are the grammatical statements and parsing, as described in the following sections, relies then on constraint satisfaction.

1.3.1 CONSTRAINTS ON TREES: ACTIVE CONSTRAINTS AND PARSING

Denys Duchier in several works with different colleagues has explored the question of constraints on trees, their representation and their implementation into a constraint satisfaction problem (Duchier and Thater, 1999; Duchier, 2000; Duchier and Debusmann, 2001). He proposed in particular a specification of dominance constraints with set operators, with the following

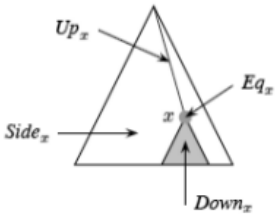
abstract syntax (adaptation from (Koller and Niehren, 2002)):

$$\varphi ::= X : f(X1,...,Xn) \mid X \lhd^* Y \mid X \perp Y \mid X \neq Y \mid \varphi \cup \varphi' \qquad (1.3)$$

In this representation, the variables X and Y denote nodes in a tree. A *dominance constraint* φ is a conjunction of different constraints shown above: labelling $X : f(X1,...,Xn)$, dominance $X \lhd^* Y$, inequality $X \neq Y$, and disjointedness $X \perp Y$.

As described by Koller and Niehren (2002): a labelling constraint expresses that X denotes a node which has the label f and whose children are denoted by $X1,...,Xn$. A dominance constraint $X \lhd^* Y$ expresses that X denotes a node that is somewhere above (or equal to) the node denoted by Y in the tree. An inequality constraint expresses that the two variables denote different nodes, and a disjointedness constraint $X \perp Y$ expresses that neither of the two nodes dominates the other.

The proposal consists then to encode this information in terms of set constraints. At each node, it is possible to identify different regions in the tree with which constraints will be expressed: the node itself , all nodes above, all nodes below, and all nodes to the side (i.e. in disjoint subtrees). These regions are denoted with different variables, respectively: $Eq_x, Up_x, Down_x, Side_x$:



Other set variables can be defined in terms of set operations, for example:

$$\begin{aligned} Eqdown_x &= Eq_x \uplus Down_x \\ Equp_x &= Eq_x \uplus Up_x \end{aligned} \qquad (1.4)$$

Constraints are then encoded by means of these set variables as in the following:

$$\begin{aligned} x \lhd^+ y \equiv \quad & Eqdown_y \subseteq Down_x \\ \wedge \quad & Equp_x \subseteq Up_y \\ \wedge \quad & Side_x \subseteq Side_y \end{aligned} \qquad (1.5)$$

This constraint stipulates that variables equal or below y are below x, variables equal or above x are above y, and variables disjoint from x are also disjoint from y. All other types of constraints on trees can be encoded as constraints on finite set in the same manner. Parsing becomes then a constraint satisfaction problem. Finding a parse consists in finding an assignment of the different variables that satisfies these constraints.

This approach of constraints applied to trees makes it clear that parsing can be seen as a constraint satisfaction problem. In this proposal, the approach can be applied to simple phrase-structure grammars and has also been proposed for dependency grammars.

1.3.2 GPSG: THE SEPARATION OF INFORMATION

GPSG (Gazdar *et al.*, 1985) was the first theory that proposed to distinguish different types of information, encoded implicitly in a tree: dominance and linearity. Dominance is encoded in GPSG with immediate dominance rules (or ID rules) and linearity with linear precedence statements (LP rules). The following example proposes a partial description of the NP in French:

$$\begin{array}{c} NP \rightarrow_{id} N; Det; \{AP; PP\} \\ Det \prec *; N \prec PP; AP \prec PP \end{array} \tag{1.6}$$

The ID-rule indicates the possible constituents of the NP, two of them being mandatory (Det and N), the others compulsory. The LP rules specify the different possible positions of the constituents in a well-formed structure. Both types of rules makes it possible to generate the set of well-formed trees. In this interpretation, ID-rules define the domain, which is the set of all possible trees, and LP-rules act as constraints on this domain, restricting the space of possible solutions to the set of trees satisfying the LP constraints. This is then a typical application of constraints, playing the role of a filtering device.

Moreover, GPSG was also one of the first theories to propose an intensive use of feature structures, enabling to encode precise relations between categories and features. GPSG was then a pioneer theory in bringing the unification process into the description of well-formedness. This was done thanks to different principles, propagating feature values through the tree nodes, but also with a specific mechanism called feature co-occurrence restriction (FCR, see also (Petersen and Kilbury, 2009)), as illustrated in the following example:

$$[+INV] \supset [+AUX, FIN] \tag{1.7}$$

This FCR stipulates that if a verb occurs initially in a sentence containing a subject (feature *[+INV]*), then this verb must be a finite auxiliary. FCRs offer the possibility to stipulate a new type of constraints. Unlike constraints on the tree structure as shown above, these constraints enables to specify relations at a fine level, directly between feature values, which is one of the basis of lexicalist approaches.

These two major innovations, ID/LP representation and constraints on feature values, are of deep importance in linguistic theory: they help in understanding what kind of information is used during the derivation process. Applying a phrase-structure rule comes then to apply a complex mechanism, relying on complex and heterogeneous information. And this kind of information (in particular LP statements and FCRs) typically acts as constraints on the structure. A constraint-based approach to GPSG parsing (see (Blache, 1992)) can then be seen as generating the set of possible trees and reducing this domain by applying the different constraints.

1.3.3 HPSG: THE NOTION OF SATISFACTION

The idea of considering parsing as a constraint satisfaction problem appeared only after constraints were introduced in linguistic theories. This question has been explored from the first-order logic point of view by Johnson (1988, 1994) that describes how "*grammaticality of an utterance corresponds to the satisfiability of a set of constraints, and ungrammaticality or ill-formedness corresponds to the unsatisfiability of that set*". This is exactly what has been applied in the theoretical elaboration of HPSG (Pollard and Sag, 1994; Sag *et al.*, 2003).

HPSG pushed one step forward the ideas of GPSG in abandoning the notion of derivation rule and representing all information and syntactic relations by means of feature value propagation. Some schema indicate the general hierarchical syntactic structure (which more or less corresponds to abstract trees) starting from which all constraints can be applied. More precisely, information being encoded by means of feature structures (hereafter FS), constraints are stipulated in terms of partial FS, each encoding a specific information. Such partial FSs are called *descriptions* and stipulate different kinds of information such as agreement, restrictions, propagation by means of structure sharing, etc.

Description application follows the attribute-value logic described by Carpenter (1992). AV-logic proposes a formal definition of feature structures together with a description language: in this language, well-formed formulas are the descriptions. All descriptions specify a particular proper-

ties of features structures. A description corresponds then to a constraint on feature structures. A satisfaction relation is then applied between feature structures and descriptions. From an operational point of view, a description implements specific relations between feature values (such as lexical restrictions) as well as syntactic principles.

Description Relations : In addition to the classical logical connectives $\vee$ and $\wedge$, AV-logic introduces four relations noted as $\delta, \doteq, :$ and @. They can be described as follow :

- $\delta(f,q)$: feature value function, expresses the value from the node q following the path f.
- $F@\pi$: expresses the value of the feature structure F at the path π.
- $\pi_1 \doteq \pi_2$: means that the value of the object got by following the path π_1 is token identical to the object got by following the path π_2.
- $\pi : \phi$: means that the value at the path π satisfies the description ϕ.

Satisfaction The following properties are not a complete definition of satisfaction relation but the interpretation of the previous relations :

- $F \models \pi : \phi$ if $F@\pi \models \pi$
- $F \models \pi_1 \doteq \pi_2$ if $\delta(\pi_1,q) = \delta(\pi_2,q)$
- $F \models \phi \wedge \psi$ if $F \models \phi$ and $F \models \psi$
- $F \models \phi \vee \psi$ if $F \models \phi$ or $F \models \psi$

Figure 1.1 illustrates this satisfaction relation. In this case, the feature structure F satisfies the description :

$$F \models (Subj \doteq Pred\ Agt) \wedge (Pred : (Pat : John))$$

Descriptions can be more or less specified according to the described properties: in this example, the description focuses on the predicative structure, but could also involve any other feature. What is important in the HPSG perspective is that all linguistic properties are described in terms of descriptions or, in other words, in terms of constraints on the structure. HPSG still relies on a phrase-structure backbone: very general hierarchical structures are defined by means of ID-schema (which are abstract trees). The constraints are applied on the structures built by combining these schema. In this sense, there is no classical derivation as in other phrase-structure

$$
\mathrm{F} = \left[\begin{array}{l}
\textsc{Sentence} : \textit{John loves Mary} \\
\textsc{Subj}: \left[\ \boxed{1}\textit{John}\ \right] \\
\textsc{Obj}: \left[\ \boxed{2}\textit{Mary}\ \right] \\
\textsc{Pred}: \left[\begin{array}{l}
\textsc{Rel} : \textit{loves} \\
\textsc{Agt}: \left[\ \boxed{1}\ \right] \\
\textsc{Pat}: \left[\ \boxed{2}\ \right]
\end{array}\right]
\end{array}\right]
$$

Figure 1.1: Example of a feature-structure satisfying an input description

approaches (for example GPSG in which ID-rules need to be applied), the structure is directly built thanks to a satisfaction process. We can say that HPSG is then a truly constraint-based theory, in which constraints are not only a filtering device, but the core of the process.

1.3.4 OT: RELAXING CONSTRAINTS

Optimality Theory (Prince and Smolensky, 1993) also occupies a central position in the use of constraints in linguistic theory. More precisely, OT introduced among other things the notion of constraint violation: constraints stipulate any kind of linguistic information, but they all can be violated by some structures.

OT has a specific architecture relying on two steps:

1. Generation (GEN): starting from an input representing an underlying form, generation of all the possible outputs, also called candidates.
2. Evaluation (EVAL): selection of the optimal candidate (the output) thanks to a set of ordered constraints (in some presentations, the set of constraints are considered as a component of the theory, as GEN and EVAL).

$$
\begin{array}{lccccc}
 & & \text{input} & & & \\
\text{(GEN)} & & \downarrow & & & \\
 & \{\ \text{candidate}_1, & \text{candidate}_2, & \ldots, & \text{candidate}_n\ \} & \\
\text{(EVAL)} & & \downarrow & & & \\
 & & \text{output} & & &
\end{array}
$$

In OT, the lexicon contains the underlying forms, starting from which GEN generates all candidates. The set of candidates is then filtered (and or-

dered) by evaluating constraints. In OT, constraints are universal: all structures should satisfy them. But at the same time, as said above, they can be violated. Two types of constraints are used: faithfulness and markedness constraints. The first require that the form of the output has to be as close as possible as that of the input. In particular, segments (or constituents) of the input need to have correspondents in the output. Moreover, the output need to keep the linear order of the input segments. Markedness constraints indicate whether the corresponding information is used to mark a variation into a given language (unmarked information being in some sense a default value). Markedness constraints indicate how the structure of the underlying form can be changed (there is then a tension between the two types of constraints). The following table illustrates the two types of constraints in phonology:

Markedness	ONSET: syllables must have an onset VOICED-CODA: obstruents must not be voiced in syllable coda position
Faithfulness	IDENT-IO: the specification of the feature [*voiced*] of an input segment must be preserved in its output correspondent

Constraints are ranked by order of importance. In OT (at least in theory) all constraints are universal. The difference between languages is taken into account thanks to different constraint orderings. In the previous example, the constraint ranking for German is the following:

$$\text{VOICED-CODA} \gg \text{IDENT-IO}$$

This ranking explains final devoicing in German (obstruents at the end of a syllable must be devoiced). For the example, the singular form of the word "*Hand*" is pronounced /hant/ while the plural "*Hände*" is pronounced /hɛnde/. The underlying form of the word (in the lexicon) is /hand/. Starting from this input, the GEN function generates many different candidates among which /hand/ and /hant/:

$$\text{GEN}(/\text{hand}/) = \{/\text{hand}/, /\text{hant}/\}$$

We can see that the form /hant/ satisfies the VOICED-CODA constraint, but violates the IDENT-IO: the input segment /d/, which is voiced, is realised as /t/ in the candidate form. The satisfaction and violation of the constraints can be summarised in a table :

Input : /bɛd/	VOICED-CODA	IDENT-IO
(a) ☞ /hant/		*
(b) /hand/	*!	