MACQUARIE UNIVERSITY, SYDNEY, AUSTRALIA
Centre for Language Technology
Department of Computing
Division of Information and Communication Sciences

and

UNIVERSITÉ DE PROVENCE, FRANCE
U.F.R. M.I.M.
ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE E.D. 184
Laboratoire Parole et Langage

Modelling Syntactic Gradience with Loose Constraint-based Parsing

≬

Modélisation de la gradience syntaxique par analyse relâchée à base de contraintes

Jean-Philippe PROST

Submitted in Partial Fulfilment of Joint Institutional Requirements for the Double-badged Degree of

Doctor of Philosophy

and

Docteur de l'Université de Provence
Spécialité : Informatique

October 2008

UNIVERSITÉ DE PROVENCE, FRANCE
U.F.R. M.I.M.
ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE E.D. 184
Laboratoire Parole et Langage

et

MACQUARIE UNIVERSITY, SYDNEY, AUSTRALIA
Centre for Language Technology
Department of Computing
Division of Information and Communication Sciences

THÈSE en COTUTELLE

présentée pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE PROVENCE
Spécialité : Informatique
et
DOCTOR OF PHILOSOPHY

Modélisation de la gradience syntaxique par analyse relâchée à base de contraintes

Modelling Syntactic Gradience with Loose Constraint-based Parsing

Jean-Philippe PROST

soutenue publiquement le 10 décembre 2008

JURY

Pr. Alexis NASR	Université de la Méditerranée	*Président du Jury*
Pr. Denys DUCHIER	Université d'Orléans	*Rapporteur*
Dr. Gerald PENN	University of Toronto	*Rapporteur*
Dr. Eric de la CLERGERIE	INRIA	*Examinateur*
Dr. Philippe BLACHE	Université de Provence	*Co-directeur*
Dr. Diego MOLLÁ ALIOD	Macquarie University	*Co-directeur*
Dr. Mark DRAS	Macquarie University	*Directeur adjoint*

Abstract

The grammaticality of a sentence has conventionally been treated in a binary way: either a sentence is grammatical or not. A growing body of work, however, focuses on studying intermediate levels of acceptability, sometimes referred to as *gradience*. To date, the bulk of this work has concerned itself with the exploration of human assessments of syntactic gradience. This dissertation explores the possibility to build a robust computational model that accords with these human judgements.

We suggest that the concepts of *Intersective Gradience* and *Subsective Gradience* introduced by Aarts for modelling graded judgements be extended to cover deviant language. Under such a new model, the problem then raised by gradience is to classify an utterance as a member of a specific category according to its syntactic characteristics. More specifically, we extend Intersective Gradience (IG) so that it is concerned with choosing the most suitable syntactic structure for an utterance among a set of candidates, while Subsective Gradience (SG) is extended to be concerned with calculating to what extent the chosen syntactic structure is typical from the category at stake. IG is addressed in relying on a criterion of optimality, while SG is addressed in rating an utterance according to its grammatical acceptability. As for the required syntactic characteristics, which serve as features for classifying an utterance, our investigation of different frameworks for representing the syntax of natural language shows that they can easily be represented in Model-Theoretic Syntax; we choose to use Property Grammars (PG), which offers to model the *characterisation* of an utterance. We present here a fully automated solution for modelling syntactic gradience, which characterises any well formed or ill formed input sentence, generates an optimal parse for it, then rates the utterance according to its grammatical acceptability.

Through the development of such a new model of gradience, the main contribution of this work is three-fold.

First, we specify a model-theoretic logical framework for PG, which bridges the gap observed in the existing formalisation regarding the constraint satisfaction and constraint relaxation mechanisms, and how they relate to the projection of a category during the parsing process. This new framework introduces the notion of *loose satisfaction*, along with a formulation in first-order logic, which enables reasoning about the characterisation of an utterance.

Second, we present our implementation of Loose Satisfaction Chart Parsing (LSCP), a dynamic programming approach based on the above mechanisms, which is proven to always find the full parse of optimal merit. Although it shows a high theoretical worst time complexity, it performs sufficiently well with the help of heuristics to let us experiment with our model of gradience.

And third, after postulating that human acceptability judgements can be predicted by factors derivable from LSCP, we present a numeric model for rating an utterance according to its syntactic gradience. We measure a good correlation with grammatical acceptability by human judgements. Moreover, the model turns out to outperform an existing one discussed in the literature, which was experimented with parses generated manually.

Keywords Gradience, acceptability, grammaticality, optimality, configuration, Model-Theoretic Syntax, Property Grammars, characterisation, constraint-based chart parsing, robustness, loose constraint satisfaction.

Résumé

La grammaticalité d'une phrase est habituellement conçue comme une notion binaire : une phrase est soit grammaticale, soit agrammaticale. Cependant, bon nombre de travaux se penchent de plus en plus sur l'étude de degrés d'acceptabilité intermédiaires, auxquels le terme de *gradience* fait parfois référence. À ce jour, la majorité de ces travaux s'est concentrée sur l'étude de l'évalution humaine de la gradience syntaxique. Cette étude explore la possibilité de construire un modèle robuste qui s'accorde avec ces jugements humains.

Nous suggérons d'élargir au langage mal formé les concepts de *Gradience Intersective* et de *Gradience Subsective*, proposés par Aarts pour la modélisation de jugements graduels. Selon ce nouveau modèle, le problème que soulève la gradience concerne la classification d'un énoncé dans une catégorie particulière, selon des critères basés sur les caractéristiques syntaxiques de l'énoncé. Nous nous attachons à étendre la notion de Gradience Intersective (GI) afin qu'elle concerne le choix de la meilleure solution parmi un ensemble de candidats, et celle de Gradience Subsective (GS) pour qu'elle concerne le calcul du degré de typicité de cette structure au sein de sa catégorie. La GI est alors modélisée à l'aide d'un critère d'optimalité, tandis que la GS est modélisée par le calcul d'un degré d'acceptabilité grammaticale. Quant aux caractéristiques syntaxiques requises pour permettre de classer un énoncé, notre étude de différents cadres de représentation pour la syntaxe du langage naturel montre qu'elles peuvent aisément être représentées dans un cadre de syntaxe modèle-théorique (*Model-Theoretic Syntax*). Nous optons pour l'utilisation des Grammaires de Propriétés (GP), qui offrent, précisément, la possibilité de modéliser la *caractérisation* d'un énoncé. Nous présentons ici une solution entièrement automatisée pour la modélisation de la gradience syntaxique, qui procède de la caractérisation d'une phrase bien ou mal formée, de la génération d'un arbre syntaxique optimal, et du calcul d'un degré d'acceptabilité grammaticale pour l'énoncé.

À travers le développement de ce nouveau modèle, la contribution de ce travail comporte trois volets.

Premièrement, nous spécifions un système logique pour les GP qui permet la révision de sa formalisation sous l'angle de la théorie des modèles. Il s'attache notamment à formaliser les mécanismes de satisfaction et de relâche de contraintes mis en œuvre dans les GP, ainsi que la façon dont ils permettent la projection d'une catégorie lors du processus d'analyse. Ce nouveau système introduit la notion de *satisfaction relâchée*, et une formulation en logique du premier ordre permettant de raisonner au sujet d'un énoncé.

Deuxièmement, nous présentons notre implantation du processus d'*analyse syntaxique relâchée à base de contraintes* (*Loose Satisfaction Chart Parsing*, ou LSCP), dont nous prouvons qu'elle génère toujours une analyse syntaxique complète et optimale. Cette approche est basée sur une technique de programmation dynamique (*dynamic programming*), ainsi que sur les mécanismes décrits ci-dessus. Bien que d'une complexité élevée, cette solution algorithmique présente des performances suffisantes pour nous permettre d'expérimenter notre modèle de gradience.

Et troisièmement, après avoir postulé que la prédiction de jugements humains d'acceptabilité peut se baser sur des facteurs dérivés de la LSCP, nous présentons un modèle numérique pour l'estimaton du degré d'acceptabilité grammaticale d'un énoncé. Nous mesurons une bonne corrélation de ces scores avec des jugements humains d'acceptabilité grammaticale. Qui plus est, notre modèle s'avère obtenir de meilleures performances que celles obtenues par un modèle préexistant que nous utilisons comme référence, et qui, quant à lui, a été expérimenté à l'aide d'analyses syntaxiques générées manuellement.

Mots-clés Gradience, acceptabilité, grammaticalité, optimalité, configuration, syntaxe modèle-théorique (*Model-Theoretic Syntax*), Grammaires de Propriétés, analyse syntaxique tabulaire par contraintes, robustesse, satisfaction relâchée de contraintes.

Contents

List of Tables

List of Figures

List of Algorithms

Preface

The research presented in this thesis is the original work of the author except where otherwise indicated. Some parts of the thesis include revised versions of published papers. This work has not been submitted for a higher degree to any other University or Institution than Macquarie University (Australia) and Université de Provence (France).

Jean-Philippe Prost

Acknowledgements

Foremost, I wish to thank my supervisors Philippe Blache from Université de Provence, and Diego Mollá Aliod and Mark Dras from Macquarie University for their trust, their encouragement, their constant support, their valuable feedback, and their unnumerable advice.

I would also like to thank the members of the examination committee for agreeing to review my work and for their insightful comments: Denys Duchier, Gerald Penn, and Eric Villemonte de la Clergerie.

This work was supported by an international Macquarie University Research Scholarship (iMURS), and by various other travel and visiting grants from Macquarie University, Université de Provence, and Ambassade de France en Australie (*French Embassy in Australia*).

I was lucky enough, during the entire course of this project, to call *home* two different countries and two different institutions, which gave me the great opportunity to meet a very large number of people from different backgrounds. They all had a considerable influence on this work, in one way or another.

From the Australian Connection, special thanks must go to Robert Dale for his always amazing wisdom. I would also like to thank people from Macquarie University and from the Centre for Language Technology: Agnieszka Baginska, Sarah Bedford, Steve Cassidy, Christophe Doche, Frederic Ehrler, Dominique Estival, Maree Graham, Mark Lauer, Joanne Pizzato, Luiz Pizzato, Vanessa Long, Cécile Paris, Brett Powley, Matt Roberts, Rolf Schwitter, Tony Sloane, Kate Stefanov, Dom Verity, and Menno Van Zaanen. I am particularly indebted to my fellow Markists, who made our weekly Reading Group meetings a necessary niche of sanity, humour and wit during The Big Journey: Elena Akhmatova, Mary Gardiner, Ben Hutchinson, Andrew Lampert, Pawel Mazur, Jette Viethen, Stephen Wan, Simon Zwarts, and obviously Mark Dras.

From the French Connection, I am also especially grateful to all the anonymous annotators involved in parts of this work for their time and expertise, as well as to a number of people from *Laboratoire Parole et Langage* for their help and useful feedback: Emmanuel Bellengier, Céline De Looze, Françoise Douay, Gaëlle Ferré, Marie-Laure Guénot, Barbara Hemforth, Cécile Petitjean, Cristel Portes, Tristan VanRullen, and Stéphane Rauzy.

This work also owes a debt to all the volunteers who kindly and courageously agreed to proofread earlier versions of this dissertation: Gail Sinclair, Matt Roberts, Ben Hutchinson, David Wren, Jette Viethen, Jason Malae, Sarah Bedford, Marc Tilbrook and Barbara Mifsud, as well as to various other people for fruitful discussions and comments, and for reviewing parts of this work: Henning Christiansen, Veronica Dahl, Ted Gibson, and Eric Würbel.

A very special mention goes to mum, encouraging and supportive as ever, who certainly did not expect having me back home with my mood and obstination after so long! I certainly do not forget either Anne-Lise, Chloé, Jean-Marc, and Véronique, and I want to warmly thank them here for their unconditional and incredible understanding and support.

Remerciements

Avant tout, je souhaite remercier mes directeurs de thèse, Philippe Blache de l'Université de Provence, ainsi que Diego Mollá Aliod et Mark Dras de Macquarie University, pour leur confiance, leurs encouragements, leur soutien permanent, et pour la pertinence de leurs innombrables commentaires et remarques.

Je souhaite également remercier les membres du jury pour avoir accepté de corriger mon travail, et pour leurs commentaires experts: Denys Duchier, Gerald Penn et Eric Villemonte de la Clergerie.

Ces travaux ont été supportés financièrement par une allocation internationale de recherche de Macquarie University (iMURS), ainsi que par diverses autres subventions pour déplacements et visites de la part de Macquarie, de l'Université de Provence, et de l'Ambassade de France en Australie.

J'ai eu le privilège, tout au long de ce projet, de me sentir chez moi dans deux pays différents, et deux institutions différentes, ce qui m'a donné l'occasion de faire un très grand nombe de rencontres, aussi variées qu'enrichissantes. Toutes ont eu, d'une façon ou d'une autre, une influence considérable sur ce travail.

Du cté australien, une mention spéciale à l'adresse de Robert Dale pour sa surprenante érudition. Je souhaite également remercier les gens de Macquarie University et du *Centre for Language Technology*, tout particulièrement : Agnieszka Baginska, Sarah Bedford, Steve Cassidy, Christophe Doche, Frederic Ehrler, Dominique Estival, Maree Graham, Mark Lauer, Joanne Pizzato, Luiz Pizzato, Vanessa Long, Cécile Paris, Brett Powley, Matt Roberts, Rolf Schwitter, Tony Sloane, Kate Stefanov, Dom Verity, et Menno Van Zaanen. J'ai une dette toute particulière envers tous les Markistes, qui ont fait de nos réunions hebdomadaires une source nécessaire de revitalisation morale, d'humour et d'esprit tout au long de la Grande Aventure : Elena Akhmatova, Mary Gardiner, Ben Hutchinson, Andrew Lampert, Pawel Mazur, Jette Viethen, Stephen Wan, Simon Zwarts, et bien évidemment Mark Dras.

Du cté français, je suis tout particulièrement reconnaissant envers tous les annotateurs anonymes qui ont participé à cette étude. Leur temps et leur expertise m'ont été précieux. Je remercie également les membres du Laboratoire Parole et Langage pour leur aide et leurs commentaires forts utiles : Emmanuel Bellengier, Céline De Looze, Françoise Douay, Ga´elle Ferré, Marie-Laure Guénot, Barbara Hemforth, Cécile Petitjean, Cristel Portes, Tristan VanRullen, et Stéphane Rauzy.

Ce travail a également une dette non-négligeable envers tous les volontaires qui ont si gentiment et si courageusement accepté de relire différentes versions préliminaires de cette dissertation : Gail Sinclair, Matt Roberts, Ben Hutchinson, David Wren, Jette Viethen, Jason Malae, Sarah Bedford, Marc Tilbrook and Barbara Mifsud. Je remercie également Henning Christiansen, Veronica Dahl, Ted Gibson, et Eric Würbel

pour diverses discussions et commentaires très utiles, et pour avoir acceptép de se pencher sur certaines parties de ce travail.

Enfin, une mention toute spéciale à ma mère dont les encouragements et le soutien sont restés sans faille, et qui ne s'attendait certainement pas à devoir à nouveau héberger mes humeurs et mon obstination après si longtemps ! Je n'oublie, bien sûr, pas non plus Anne-Lise, Chloé, Jean-Marc, et Véronique, que je veux remercier ici pour leur compréhension, et leur incroyable et inconditionnel soutien.

Uno lengo es un clapas ; es uno antico foundamento ounte chasque passant a tra sa pèço d'or o d'argènt o de couire ; es un mounumen inmènse ounte chasco famiho a carreja sa pèiro, ounte chasco ciéuta a basti soun pieloun, ounte uno raço entiero a travaia de cors e d'amo pendènt de cènt e de milo an.

Uno lengo, en un mot, es la revelacioun de la vido vidanto, la manifestacioun de la pensado umano, l'estrumen subre-sant di civilisacioun e lou testamen parlant di soucieta morto o vivo.

(Frederi Mistrau, 1877. *La lengo dóu Miejour*. Discours à l'Assemblado de Santo Estello d'Avignoun, 21 de Mai 1877)

Une langue est un bloc : c'est un antique fondement où chaque passant a jeté sa pièce d'or, d'argent ou de cuivre : c'est un monument immense où chaque cité a bâti son pilier, où une race entière a travaillé de corps et d'âme pendant des centaines et des milliers d'années.

Une langue, en un mot, est la révélation de toute une vie, la manifestation de la pensée humaine, l'instrument sacro-saint des civilisations et le testament parlant des sociétés mortes ou vivantes.

(Translated from Prouvencau by the *Lexilogos* web site,
`http://www.lexilogos.com/provencal_mistral_discours.htm`,
as of 22 January 2008)

A language is a block: it is an antic fundament where every passer-by laid their gold, silver or copper coin: it is an immense monument where every city built its pillar, where an entire race worked from their body and soul during hundreds and thousands of years.

A language, in a word, is the revelation of a lifetime, the demonstration of human thought, the sacro-sanct instrument of civilisations and the talking legacy from living or dead societies.

(Personal translation)

Chapter 1

Introduction

In Computational Linguistics, grammaticality has traditionally been treated as a binary notion, whereby a string either does or does not belong to language. Yet in practice it has also long been admitted that uttered language, from gibberish to classical literature, reveals intermediate degrees of acceptability.

Intuitively, motivations for studying graded acceptability are easy to grasp. We have all experienced everyday-life situations where uttered language may be more or less acceptable, whether spoken or written. Subsequently, handling these situations automatically is a problem faced in various domains of Language Technology (LT) such as Text Analytics[1], Information Retrieval, Summarisation, Machine Translation, Question-Answering, Natural Language Generation, and so on. Just as an example—and to take cases of not-so-intuitive situations, *grammaticality measures* may be used in Summarisation (Wan et al., 2005) in order to evaluate qualitative aspects of language. In Machine Translation as well, alternative sentences in the target language may be discriminated on the basis of their measured grammaticality. However, very few studies from Computational Linguistics have attempted to systematically investigate the theoretical possibility of capturing a gradient of acceptability with a computational model, and evaluate to what extent model and human judgement correlate.

[1] The term *Text Analytics* appeared fairly recently in the LT community, and covers the fields previously known as Text Mining or Information Extraction. See the Text Analytics Wiki (`http://textanalytics.wikidot.com/`) for more information.

The venture is indeed challenging in many respects. Firstly, the observation of the phenomenon to be modelled is arguable in nature since it concerns human judgement, hence it is intrinsically subjective; evaluating the fit of a model is, therefore, open to argument. The lack of a reference is a serious impediment to providing a scale of magnitude for numerical estimates. Secondly, because the knowledge involved in making such a judgement being itself fairly unclear, all sorts of problems regarding its representation inevitably arise. What kind of information is involved in a judgement about acceptability? How to represent that information? How to process it in order to make a decision? And thirdly, what are the numeric elements which a predictive model of grammatical acceptability should be based upon? Can all linguistic phenomena involving syntactic gradience be captured by the same numeric account, or do they require different numeric models to be designed?

This dissertation explores the possibility to build a robust computational model that correlates with human judgements. We address the questions above in order to provide a numeric model, which captures the graded aspects of the syntax of a language—referred to as (syntactic) *gradience*. The problem is addressed from the two angles of knowledge representation and numeric modelling.

This work is organised around four main parts. In Chapter 2, taking as a starting point existing linguistic and psycholinguistic analyses of gradience, we investigate (a subset of) them with respect to the syntactic information they contain, and how to present, represent, and process it. We explore different attempts made by others to circumscribe linguistic phenomena involving gradience and we draw conclusions about the type of linguistic knowledge required. The model of syntactic gradience devised by Aarts (2007) is of particular interest, even though it is not concerned with ill formed language. The proposed dichotomy of phenomena involving gradience into those concerned with *Intersective Gradience* and those concerned with *Subsective Gradience* is identified to be particularly well-suited to serve as a bottom line for further extensions, in order to cover other dimensions of language than those tackled by Aarts. We also consider different existing options for adequately representing that linguistic knowledge, by making sure that the representation we choose suits the purpose of a numerical assessment. Hence we explore the ability of various computational frameworks to overcome syntactic deviance and still inform on observed syntactic characteristics. On the processing side, we explore various parsing strategies, which are compatible with the kind of representation of language as previously identified, and which present the resulting linguistic information in a form suitable to numerical processing. Our investigation leads us to choosing Property Grammars (PG) (Blache, 2000, 2005) as one of the most suitable frameworks.

Yet this framework presents a number of shortcomings in its existing formalisation, which prevent from reasoning with the *characterisation* of an utterance—a key concept introduced in PG for reprensenting the syntax of an utterance. We observe that the problem does not come from the theory in itself as presented initially, but more from its formalisation (VanRullen, Guénot, and Bellengier, 2003; VanRullen, 2005), which takes a different point of view and does not aim to address the question of reasoning. Therefore, in Chapter 3 we specify a new logical system for PG, in order to further formalise the theory and overcome the issue at stake for the purpose of reasoning.

In Chapter 4, we introduce the implementation of a robust chart parser using the model-theoretic framework from the previous chapter. This parser is robust in always producing a full parse for unrestricted input. It also aims to ensure that the output analysis is the optimal one with respect to a merit function. Such an aim is theoretically an important one as it directly participates in the model of syntactic gradience we advocate through this work. We calculate the algorithm's theoretical worst time complexity, and we evaluate its performance over the same corpus as we are using for the experimental study from next chapter.

In Chapter 5, given the parser's output, we experiment with numeric models of syntactic gradience, and measure to what extent each model fits acceptability by human judgements. Relying on the reviewed literature, we start by postulating different factors of influence on acceptability which are substantiated by linguistic and/or psycholinguistic evidence. These factors are complemented by other intuitive ones, for which no linguistic studies were found in the literature. Each of these factors is captured in numerical terms, which derive from syntactic elements taken from the parser's outputs. These terms are then used in different rating functions, each one corresponding to a model of gradience. One of these models is a pre-existing one presented in Blache, Hemforth, and Rauzy (2006). We run a comparative investigation of all of them, which aims to determine which one performs better at predicting human judgement of acceptability. The gold standard we are using comes from a psycholinguistic experiment run independently from the present study. That experiment used Magnitude Estimation (Bard, Robertson, and Sorace, 1996) with human annotators in order to assess acceptability of individual sentences. The corpus is mostly made up of controlled ill-formed sentences (94% of the total). The fit of each model is figured by Linear Regression, and a correlation coefficient is measured: on the full corpus a correlation $\rho = 0.54$ is is obtained, and $\rho = 0.64$ is obtained on exactly the same data sample (from the same corpus) as used by Blache, Hemforth, and Rauzy, who report a correlation $\rho = 076$. The most salient difference between their experiment and ours is that Blache, Hemforth, and Rauzy rely on manual parses, while we experimented with syntactic parses which were automatically generated by our parser.

Chapter 6 draws conclusions and discusses avenues for further work.

Chapter 2

Background

Anyone who knows a natural language knows that some utterances are not completely well formed. Speakers produce utterances that even they would agree are grammatically imperfect — not by some external authority's standard but by their own. But experienced users of a language are also aware that some ungrammatical utterances are much closer to being grammatical than others.
(Pullum and Scholz, 2001)

2.1 Introduction

The above epigraph from Pullum and Scholz perfectly summarises the starting point of this work. *Natural language* must be taken here in its

> ordinary, common-sense notion (...) under which we can say that *The Times* in the UK, *The New York Times* in the USA, *The Sydney Morning Herald* in Australia, and other newspapers around the world, all publish in the same language — though of course we would not deny that there may be local differences concerning which expressions are judged grammatical by the relevant editors. (Pullum and Scholz, 2001, p. 38)

If we agree—and we do—that such a notion of natural language is indeed *ordinary*, then it naturally entails that studying natural language must account for language in its entirety, which includes well formed utterances from the UK, the USA, or Australia, as well as those being “not completely well formed” or “grammatically imperfect”, regardless of how close they are from being grammatical. We come back on these aspects into more details in §2.3.1. Then once acknowledged such variations, it seems reasonable to wonder about how to answer the question *how close is such or such utterance from being grammatical?*, and about what form the answer can take. These are the broad questions we have in mind for the present chapter, and more generally for this study.

We start, in §2.2, by exploring epistemological aspects of linguistic *gradience*, as the propensity of language to possess varying degrees of acceptability is sometimes referred to. In section §2.3 we explore the different options available in terms of types of theoretical frameworks for dealing with a notion of natural language which matches the ordinary one. In §2.4 we investigate more specifically the existing computational models of gradience. In §2.5 we then focus on exploring different specific frameworks involving gradience, or open to an account of it. §2.6 concludes the chapter; it summarises our review of the literature and states the standpoint we will be taking for the rest of this study.

2.2 Epistemology of Gradience

This section presents a brief overview of epistemological considerations around gradience, with a prime focus on syntactic gradience in natural language. The section is essentially—though not exclusively—based on the reader on *fuzzy grammar* from Aarts et al. (2004), as well as Aarts' own works on gradience (2004a; 2004b; 2007; May 2007). Together they cover a very large spectrum of the literature, from Aristotle until modern time, and constitute a thorough investigation of the topic. See also Haji-Abdolhosseini (2005, pp. 7–11) for another interesting review of gradience in grammar, also presented around Aarts' works (until 2004).

As a starting point and since, to the best of our knowledge, there exists no reference definition as such for the term *gradience*[1], we venture to sum up our readings with the following one:

Gradience denotes the property or fact of being variable, or graded; propensity

[1] In the English literature in linguistics, the term *Gradience* seems to appear for the first time in 1961 with Dwight Bolinger in his book *Generality, Gradience, and the All–or–None* (1961).

to variability or boundary vagueness. Applied to linguistics it refers to the propensity of natural language to possess varying degrees of acceptability.

Different views and interpretations of gradience co-exist in the literature. In very general terms, questions of gradience arise in categorisation problems, where it refers to categorical vagueness. From there, different interrogations arise: Isn't classical categorisation enough, as advocated in Bouchard (1995) and Newmeyer (2000)? Does "categorical vagueness" refer to Aritotelian sharp boundaries with strict membership rules while acknowledging the possibility for certain members to be atypical to different degrees (Jackendoff, 1983; Lakoff, 1987a; Aarts, 2007, May 2007)? Or does it refer to loose boundaries, with membership rules applying to different degrees (Ross, 1972, 1973; Lakoff, 1973, 1987a; Sorace and Keller, 2005)[2]? Should degrees of acceptability be captured on a discrete scale, or should it be a continuum (Schütze, 1996; Duffield, 2003)?

As far as categorical indeterminacy is concerned, different situations are concerned with gradience.

Unmet Membership Requirements Cases where items can not be categorised properly because there is not any class for which the item to be classified meets exactly all the membership requirements. For Hudson (1990), for instance, the question of whether a three-legged cat is still a cat is addressed by the *Best Fit Principle*.

Membership Ambiguity As the number of membership features grows, an item could belong to either one or another class, and thus assigning it to one or the other class might not be as trivial and obvious as it is when all the features are met. In this case the problem is slightly different and is now concerned with discriminating more or less precisely between more than one category membership. Aarts uses the metaphor of a heap of sand (Eubilide's Paradox of the Sorites[3])

[2] For Lakoff (1987a), some categories, like *tall man* or *red* are graded, with fuzzy boundaries, while some others, like *bird* have sharp boundaries. That explains why Lakoff is cited in both cases.

[3] Check the Standford Encyclopedia of Philosophy's entry (Hyde, Fall 2005) for more details on the Sorites Paradox.

to illustrate the resistance of an item to a change of state, from member of category A to member of category B:

> Does one grain make a heap? Evidently not. Do two grains make a heap? No. Do one hundred grains make a heap? Yes. Where does one draw the line? (...) when can we call a collection of grains a heap? Is there a cut–off point such that n grains of sand form a heap, but $n-1$ do not?

Aarts proposes an interpretation of gradience based on prototypicality. Using a set-theoretic-like terminology, two classes of problems concerned with gradience are discriminated: the phrase *Subsective Gradience* (SG) is coined for referring to "degree of resemblance to a categorial prototype", while *Intersective Gradience* (IG) is coined for referring to "degree of inter-categorial resemblance". *Subsective Gradience* has to do with how close an item is to a prototypical item within a given category, while *Intersective Gradience* has to do with the relative position of an item between two (or more) categories. Aarts investigates linguistic phenomena from that SG/IG angle. Phenomena are taken at the word level (*e.g.*, SG within the Adjective class), and at the constructional level. We only report here the discussion about the phrasal and constructional levels, as we are not concerned, in this thesis, by the lexical level.

2.2.1 Subsective Gradience

In the SG approach, members of a category may only present a subset of the features which characterise a prototypical member of the category. The degree of prototypicality of a form class member then depends on how many of these features are satisfied.

Aarts insists on the risk of confusion between 'degree of prototypicality' and 'degree of membership', the latter notion being best illustrated by a reference to Jackendoff (1983) for whom

> (...) one might think of a typical bird such as a robin as 100% bird, but a penguin as perhaps only 71% bird and a bat as 45% bird. (...) The difficulty with this view (pointed out to me by John Macnamara) is that a penguin is not 71% bird and 29% something else, it just *is* a bird.

In this regard, Aarts also claims that:

> (...) linguistic formatives, while showing degrees of form class prototypicality, do not display degrees of class membership.

Class membership is seen as a binary (*i.e.* all-or-none) notion, which requires well-defined boundaries for classes, whereas prototypicality is a graded notion. For instance, in the class of adjectives (Aarts, 2007, p. 105):

> (...) happy is a more typical (alternatively, 'prototypical') exemplar of the class of adjectives than thin, which in turn is more centrally like an adjective than alive and utter.

This basically means that *happy*, *thin*, *alive* and *utter* are all legitimate members of the class of adjectives, even though they present different degrees of prototypicality.

In that approach gradience is not ubiquitous and categories have sharp boundaries. What this point emphasises is that gradience is not about defining loose boundaries to categories, but rather about loosing the rules of categorisation. A 'gradience-is-everywhere' kind of approach, as Aarts calls it, would make a different assumption in this regard.

2.2.2 Intersective Gradience

Intersective Gradience is concerned with phenomena which show properties from two different form classes. Adopting Aarts' terminology, a form class member is said to *converge* on another class when it presents properties from both categories. The question is raised and investigated as to whether linguistic gradience is a matter of fuzzy class boundaries and intersecting categories, or whether the classes have sharp boundaries. In the latter case, advocated by the author, the classes do not intersect and the formatives concerned with (intersective) gradience are seen as a set of properties which intersect with the sets of properties from two form classes.

2.2.3 Constructional Gradience

For what we are interested in, *Constructional Gradience* (CG), as coined and disscused by Aarts, is merely a generalisation of the IG/SG view on gradience, where the classes are syntactic constructions. The notion of construction as adopted in Aarts is intentionally quite broad, and although not unrelated to Construction Grammar (CxG) (Goldberg, 1995; Kay, 2002) (see §2.3.3 for more details), no explicit connection is made to one constructionist framework in particular. Instead, a construction is described informally as

> (...) a string of elements (words, phrases, etc.) that has a distinctive patterning which plays a role in different parts of the grammar. Put differently, in an everyday and pre-theoretical sense, the term 'construction' designates a particular configuration of elements that has an identifiable grammatical role to play. These characterizations give prominence to the linear ordering of elements, and exclude semantic considerations from the definition. (Aarts, 2007, p. 170)

Syntactic Characterisation It is clear from the quotation above that Aarts adopts a syntactic approach to construction, which rules out semantics, arguing that "the semantics attributed to constructions is often too vague (...), too elaborate (...), or too skeletal" (Aarts, 2007, pp. 192–195). We think that such an assumption might turn out to be quite restrictive, in some cases. In order to illustrate these situations, it is probably best to point out the distinction made in CxG between the construction's *meaning* on one hand, and on the other hand the licensing *semantic properties*, which may participate in the construction definition. The authoritative definition of a construction in CxG (Goldberg, 1995, p. 5) states:

> C is a CONSTRUCTION *iff* C is a form-meaning pair $\langle F_i, S_i \rangle$ such that some aspect of F_i or some aspect of S_i is not strictly predictable from C's component parts or from other previously established constructions.

Meanwhile, and still in Goldberg (1995, p. 152), constructions such as the Caused-Motion Construction (C-MCx) are discussed where semantic properties are among the requirements specified in the definition of the construction, alongside the syntactic structure (emphasis is ours):

> [The English Caused-Motion] construction can be defined (in active form) structurally as follows (where V is a *nonstative verb* and OBL is a *directional phrase*:

[SUBJ [V OBJ OBL]]

This definition is meant to cover the following types of expressions:

1. They laughed the poor guy out of the room.
2. Frank sneezed the tissue off the table.
3. Mary urged Bill into the house.
4. Sue let the water out of the bathtub.

In this case the definition of the construction clearly includes semantic properties, namely a *nonstative verb* and a *directional phrase*. Follows an interesting discussion, where Goldberg investigates examplars such as 1 and 2, for which it is observed that the construction involves intransitive predicates which, out of the context of this construction, can not occur transitively at all. Therefore, analysing cases of C-MCx without taking into account these licensing semantic properties would mean dealing as well with formatives which present the same licensing syntactic properties as those of the C-MCx, but which are not C-MCxs, precisely because the licensing semantic properties do not hold.

Clearly, Aarts decides to prevent *meaning* from accounting for gradience, which is fine; but he seems to reject the *licensing semantic properties* as well, which, from our point of view, is more arguable, for the reason mentioned above. In fact, no such a disctinction is acknowledged regarding semantics. Instead, it is argued that a construction *structure* ought to be studied strictly distinctively from its *usage*, which also cover pragmatics—and, we understand, possibly more dimensions. Then in order not to rely on semantics, Aarts advocates a characterisation of constructional IG and SG "by making reference to the distributional potential of particular constructions". This way, "[t]he most typical exemplar of a construction will be the one that is distributionally the most versatile" (Aarts, 2007, p. 196). This use of the distributional properties of constructions to define form classes is challenged by Croft (May 2007), in turn argued further by Aarts (May 2007). We are not going to take a more active part in the debate, which goes beyond the scope of our work, since the discussion is mostly concerned with non-deviant formatives.

2.2.4 Markedness

Aarts (2007, p. 91) reports that the notion of *markedness* was first used in works on syntax in Chomsky (1961), but was mostly developed in the 1980s (Chomsky, 1981). Markedness comes along with the dual notions of *core* and *periphery* of a grammar. The underlying idea is that Universal Grammar specifies a core grammar, which tackles the unmarked cases, while the marked cases belong to the *periphery.* The core grammar has a fixed set of rules and parameters, which, according to Aarts,

> (...) can be conceived of as mental 'switches' which are either set to 'on' or 'off', depending on the particular language. Probably the most well-known parameter is the Head Parameter, which stipulates that languages are either 'head first' or 'head last', that is, heads either precede their complements, or follow them. English is a head-first language, Japanese is a head-last language.

As for the periphery, for Chomsky it consists of

> borrowings, historical residues, inventions, and so on.... [The] marked structures have to be learned on the basis of slender evidence...,so there should be further structure to the system outside of core grammar. We might expect that the structure of these further systems relates to the theory of core grammar by such devices as relaxing certain conditions of core grammar, processes of analogy in some sense to be made precise, and so on, though there will presumably be independent structure as well: hierarchies of accessibilities, etc.
>
> Chomsky (1981, cited in Aarts, 2007, p. 92)

Aarts points out that Generativists "envisaged a radical split between core and periphery, but allowed for some kind of gradience *within* both the core and the periphery". Markedness is thus a means for dealing with variation, but remains a binary notion. For Aarts,

> switches, after all, do not allow for mid-way positions. (...) the notions of core and periphery too are very distinct, and the theory does not appear to allow for a shading between the two. The gradient markedness model can

> be seen as a compromise between the gradient and all-or-none conceptions of the nature of grammatical architecture (although, of course, it was never intended to be like that). (Aarts, 2007, p. 93)

Aarts goes further in asserting that markedness can not be seen as the same as subsective gradience, since "[m]arkedness divides categories into two, a core and a periphery, whereas subsective gradience recognizes a gradient amongst elements within categories".

2.3 Gradience and General Frameworks for Knowledge Representation

2.3.1 Generative-Enumerative Syntax *vs.* Model-Theoretic Syntax

The question now arises as to what kind of formal framework should be chosen in order to state grammar(s) and study the syntax of natural language in a fashion which allows for representing and dealing with gradience. As pointed out by Pullum (2007),

> work in formal syntax over the past fifty years has been entirely dominated by (...) the string-manipulating combinatorial systems categorized as *Generative-Enumerative Syntax* in Pullum and Scholz (2001).

Generative-Enumerative Syntax (GES) models a language as a set of legal strings. A GES grammar provides a set of production rules which enables the enumeration of all the elements in this set. In fact, and more precisely, the problem of enumerating all legal sentences in the language is seen as being equivalent to the problem of generating a tree structure for each of them, the vocabulary being the given finite set of terminals. The enumeration is then performed in combining the grammar rules in all the possible ways in order to generate the set of equivalent trees. When the process succeeds and generates a tree the sentence being analysed, seen as a set of terminals, is said legal according to the grammar in use.

Such a reading of the parsing process in GES will allow us to make a few points. One is with regard to the vocabulary: the lexicon of the natural language at stake and the lexicon of terminals for the meta-language used to describe the equivalent tree structures are made one and same thing. That is to say, the lexicon used by the meta-language to describe natural language is the same as the lexicon of the natural language being described. Practically it means that the natural words—or characters if we take the problem from the morphological level—make up the set of terminals of the description language. As a consequence, the theory is incapable of handling the dynamic aspect of natural language whereby new lexical items are created every day—and this, beyond the trivial fact that creating an exhaustive list of all the lexical terms in a natural language is quite impractical. It disregards *lexical openness* as a legitimate property of natural language.

Another point is with regard to the grammar rules: a grammar rule is a production rule, which merely informs on what the tree structure of an enumerable sentence is. There is no room for informing on the grammatical properties that hold or fail for a sentence which can not be enumerated. This aspect of GES closes the door to providing a grammatical analysis of a so-called ill-formed sentence. If no sequence of grammar rules can be found to generate a string then very little can be said about the syntactic structure of the input sentence, besides maybe leaving behind a series of alternative partial sequences.

As a matter of fact, a body of work aims to recover from the failing process and keep the enumeration going in relying on these partial sequences. Various techniques have been deployed to extend GES and develop recovery processes. Weischedel and Sondheimer (1983), for instance, use grammar-dependent recovery *meta-rules* in order to diagnose errors and try to generate parses of longest path for ill-formed input; Carbonell and Hayes (1983) use a similar approach to focus on specific types of errors; Mellish (1989) relies on grammar-independent extra rules to exploit the information left behind by an active chart parser and to search the space of incomplete parsing parses for sources of errors; McCoy, Pennington, and Suri's *mal-rules* (1996) also expand the grammar with a set of error productions; mal-rules are also found more recently in Bender et al.'s system for Computer-Assisted Language Learning (2004), or declined as an *error grammar* in Foster and Vogel (2004). In a similar yet slightly different trend, Douglas and Dale (1992) propose a mechanism of *constraint relaxation* to extend the PATR-II formalism. However, all these approaches are limited in scope as to which extra-grammatical linguistic phenomena can be diagnosed or recovered from, and most of them are grammar-dependent. Therefore a solution still needs to be found, which would abstract away from specific errors and specific parsing strategies, in order to analyse unrestricted natural language and report on gradient grammaticality. GES models a language as a set of legal strings, and anything which does not belong to that set is simply ruled out from the domain of study.

It raises the question of whether natural language, as an "ordinary, common sense notion", can be covered in all its aspects by such a formal framework. The question was addressed in various works from Pullum and Scholz (2001; 2005; 2007) (henceforth P&S), who compare the family of GES frameworks and the family of *Model-Theoretic Syntactic* ones, and investigate how different linguistic phenomena can be described in each of the two frameworks. The phenomena being mostly discussed are *lexical openness*, *gradient gramaticality*, and *expression fragments*. We will come back on their argument shortly.

According to P&S the works on GES originate from Post's formalisation of proof theory (Post, 1943). In GES the notion of *language* is defined with respect to the one of *grammar*, as the set of strings which can be generated by the grammar. A grammar is said *generative* in the sense that it provides a recursive enumeration of sets. The family of GES frameworks is consider to cover "all the familiar types of phrase structure grammar, classic MIT transformational grammar (TG), Backus-Naur form, and all typical statements of the syntax of programming languages[4]", as well as "all of categorial grammar including Montagovian syntax, tree adjoining grammar, the 'minimalist program', the neominimalism of Stabler and Keenan, and nearly all

[4] This view of statements of the syntax of programming languages (PL) is a contentious point in P&S's view: thanks to Gerald Penn (examiner, *rapporteur*) for pointing out that "BNF is actually a very superficial way of describing PL syntax as strings, and is only the final step in the development of a PL syntax that begins with an 'abstract syntax' that is much more akin to Pullum's model-theoretical view. What is interesting is that the PL community do not have any problem with using both—not only are they not status competitors, but they complement each other by addressing different aspects of computation and interpretation."

statements of the formation rules for logics" (Pullum, 2007).

P&S claim that such an approach to formalising the syntax of natural language presents several drawbacks, and present *Model-Theoretic Syntax* (MTS) as an alternative "do[ing] much better on representing the facts" about different natural language phenomena. Unlike GES, "MTS takes a grammar to be a set of statements of which (algebraically idealized) well-formed expressions are models". More specifically,

> MTS frameworks, as I understand them, are distinguished by the adoption of three general positions: (I) rules are statements about expressions; (II) grammars are finite sets of such rules; (III) well-formedness of an expression consists in satisfaction of the grammar. (Pullum, 2007)

In other words, MTS takes a descriptive point of view on syntax, whereby "a grammar should describe the syntactic properties of expressions of a language by making general statements about their syntactic properties".

Expression P&S use the term *expression* to refer to the objects that linguist investigate. It covers sentences, clauses, phrases, words, idioms, lexemes, syllables, ... from natural language in its ordinary, and common sense. Expressions are taken

> to *have* syntactic structure, not merely to be analyzable in terms of structures imposed on them or posited for them by linguists. (Pullum, 2007)

P&S also coin the term *quasi-expression* to refer to an expression, which violates at least one rule.

Rules In Pullum (2007) MTS rules are defined as being:

> simply assertions about the structure of expressions. That is, an MTS rule makes a statement that is either true or false when evaluated in the structure of an expression. If a structure is to be grammatically well formed according to a certain rule, then the rule must be true as interpreted in that structure.

Such a definition for MTS rules makes them incredibly similar to the *characteristic properties* from Aarts, with perhaps the difference that Aarts seems not to have generalised the notion to quasi-expressions. In contrast, in GES a *rule* is rather a processing instruction; it is a step in the procedure of derivation, which does not state anything about the well-formedness of an expression and can not be interpreted in isolation. P&S exemplify the difference with the case of the 'Merge' operation in Transformational Grammar:

> 'Merge' cannot be understood as a condition on the structure of expressions. It is a dynamic tree-building concatenation operation, joining two items together and adding a node immediately dominating them. Notice that it is stated informally as an imperative. (Pullum, 2007).

Grammar An MTS grammar is simply a finite, unordered set of rules. Since no procedural aspect is involved, a grammar does not include any instructions or restrictions regarding a sequential application of rules. For instance, rules about verb agreement are stated independently from rules of word order. P&S give the following examples of grammatical statements: 'The subject noun phrase of a tensed clause is in the nominative case'; 'The head verb of the verb phrase in a tensed clause agrees in person and number with the subject of that clause'; 'Verbs always follow their direct objects'; or 'Attributive modifiers precede the heads that they modify'.

What is not clear, however, is whether the pairwise independence of the rules in the grammar is a strict requirement. Bearing in mind the modelling of gradience it seems important to enforce that a constraint be atomic, in the sense that it models a single grammatical statement. We are concerned here with cases where it may be necessary to subordinate the success of a constraint to (the success or the failure of) another one. One could imagine, for example, the need for a rule such as 'In French, when the direct object precedes the auxiliary *avoir* [*to have*] the past participle agrees in person and number with the direct object'. Such a rule is arguably made of two statements: a first statement 'The direct object may precede the auxiliary *avoir*', followed by a second statement, which is conditionned by the success of the first one: 'The past participle agrees in person and number with the direct object'. For the sake of gradience it might be useful to have two, or maybe three statements: the implication, its consequent, and its antecedent. When it comes to account for the

gradience of an utterance we just want to make sure that we do not count either too many or too few satisfied and/or violated statements.
We take the lack of specific requirement in that respect as flexibility provided to the grammar writer.

Grammaticality In GES an expression is *grammatical*, that is, *well-formed* if and only if it can be generated—in the sense of derivation—by a series of rules from the grammar. That is, the notion of grammaticality is a procedural one.

Alternatively in MTS,

> [a]n expression is well formed according to an MTS grammar if and only if the semantic consequences of the grammar are true in its syntactic structure. Grammaticality is thus defined by reference to the SEMANTIC consequences of rules (the semantics of the formal language in which the rules are stated, that is — not the semantics of the natural language being described). An expression is fully well formed if and only if its structure complies with every requirement that is a semantic consequence of what the grammar says. (Pullum, 2007)

Grammaticality in MTS contains no procedural aspect. Instead, the grammaticality of an expression results from the observation of its syntactic properties. Conversally, ungrammaticality in MTS is defined by violation of one or more of the rules of the grammar.

Such a notion of grammaticality is also compatible with the sharp boundaries Aarts assigns to categories. Since the notion of expression covers a rather large spectrum of items, from syllables to sentences, in case of a phrase structure the definition of grammaticality applies not only to natural language as a whole, but also to syntactic categories, from lexical to constructional. Therefore, in such an approach, categories have sharp boundaries, in that an expression can clearly be qualified as being either grammatical or ungrammatical, whether or not it violates some of the rules specifying the category.

P&S compare the two families of frameworks on their ability to describe different phenomena of natural language considered as being distinctive. They show not only that MTS is well suited to describing their syntactic structure, but also that GES is conversely ill suited to their description. The different phenomena under investigation concern:

- *gradient ungrammaticality*, that is, "the gradient character of the property of being ungrammatical";
- *expression fragments*, like interruptions in a conversation;
- *lexical openness*, that is, "the open nature of natural language lexicons".

As much as we agree we P&S that GES is no suited to describe gradient ungrammaticality and lexical openness, we would be more cautious about expression fragments, as we have seen previously that GES could be extended in various ways to handle these cases. We take P&S's view as concerning the strict GES frameworks only in this respect.

Next we report their discussion about gradient ungrammaticality, similar arguments applying to lexical openness.

Gradient Ungrammaticality P&S consider that some utterances, the *quasi-expressions*, are "vastly less deviant than others". An unaugmented GES framework can not say anything about these quasi-expressions, since no derivation can generate it—if there was one, then it would be a well-formed expression. Chomsky (1961) attempted to augment a generative grammar in order to describe degrees of ungrammaticality. P&S show why the proposed solution is inadequate. Chomsky's solution consists of mapping the ungrammatical strings from the complement of the generated set to a number representing the degree of their ungrammaticality. Given a language over a vocabulary V_T, the proposed function $f : V_T^* - L(G) \mapsto \{1, 2, 3\}$ maps any sequence w to a degree of ungrammaticality. P&S discuss a function, which relies on a relation of similarity between lexical category sequences, and emphasise three problems with the proposal. One problem is that three degrees are not sufficient, far more degrees being easily observable, and more importantly that "none of the degrees of difference in ungrammaticality in these examples is described under Chomsky's proposal (it does not cover deviance resulting from violations of constraints on inflection at all, as far as we can see)". Example 2.1 is provided where the six expressions are increasingly ungrammatical.

(2.1) a. I am the chair of my department.

b. *I are the chair of my department.

c. *Me are the chair of my department.

d. *Me are the chair of me's department.

e. *Me are chair the of me's department.

f. *Me are chair the me's department of.

That first argument might be a bit excessive, in that that the number of degrees being limited to three is more of an incidental property of Chomsky's proposal.

The second problem being stressed is "the lack of any relationship between the proposed degrees of ungrammaticality and any specific violations of grammatical constraints". The argument relies essentially on the observation that Chomsky's proposal to assign a degree to an arbitrary set of words depends only on similarities among lexical category sequences, and does not depend on any aspect of syntactic structure. As a consequence, the degree assigned to an ungrammatical string would be exactly the same for all the grammars sharing the same lexicon. By contrast, "with model-theoretic grammars the same resources employed to describe the fully grammatical expressions also yield a description of the quasi-expressions". An example is presented, which suggests that degrees of ungrammaticality be described by defining a partial order among structures. A possible solution is sketched as follows. An arbitrary set of structures for expressions and quasi-expressions is assumed (universe $\mathcal{U}$ of labelled trees), along with an MTS grammar Γ; a partial order $\lhd_{\Gamma}^{\mathcal{U}}$ is defined for $\langle \mathcal{U}, \Gamma \rangle$, which holds between two structures when one of the two is *at least as close to being grammatical* as the other. It is captured by the fact that $\tau_1 \lhd_{\Gamma}^{\mathcal{U}} \tau_2$ if and only if τ_1 satisfies at least as many of the constraints of Γ as τ_2:

$$\lhd_{\Gamma}^{\mathcal{U}} \overset{\text{def}}{=} \{\langle \tau_1, \tau_2 \rangle : \tau_1, \tau_2 \in \mathcal{U} \wedge |\{\phi : \tau_1 \models \phi\}| \geq |\{\phi : \tau_2 \models \phi\}|\} \tag{2.2}$$

$\tau_1 \lhd_{\Gamma}^{\mathcal{U}} \tau_2$ means, when true, that the quasi-expression presenting the structure τ_1 is ungrammatical to a greater degree than a quasi-expression presenting the structure τ_2.

An important point being emphasised by P&S is that "no extra machinery is called for: the suggested analysis of degrees of ungrammaticality simply exploits the content of the MTS grammar that is constructed to describe the fully grammatical expressions: on any set $\mathcal{U}$, there is a relation $\lhd_{\Gamma}^{\mathcal{U}}$ for MTS grammar Γ". Note the importance of the arbitrary set of syntactic structures $\mathcal{U}$: our understanding is that it would take more than defining a partial order to GES just to generate the equivalent of $\mathcal{U}$ and to achieve the same result, if possible.

The third problem is that there exists no algorithm for determining the status of an arbitrary word sequence, since the question[5] is undecidable because of the Turing-equivalence of Transformational Grammar.

2.3.2 Optimality Theory

Optimality Theory (OT) (Prince and Smolensky, 1993) is a linguistic theory based on principles and parameters, and is usually seen as a development of generative grammar. Although mostly used in works on phonology, OT is also a framework suitable to natural language parsing. From a processing point of view, the central idea of the theory is that an input can be described by an optimal output structure, whose choice, among a set of candidate structures, results from a competition among constraints. A grammar consists of a hierarchy of constraints on candidate structures. The constraints are *ranked*, *universal* and *violable*. Two types of constraints co-exist, for encoding either *faithfullness* or *markedness*. The constraints of faithfullness specify to what extent an output structure should fit the input. For instance, the output must—or not—preserve the linear order among segments in the input. The constraints of markedness are used to implement parameters—in the sense of Principle

[5] In plain language, the question could be worded informally as follows: *For a Transformational Grammar G, a word sequence $w \notin L(G)$, and a lexical category sequence κ for w, can we find a word sequence w' in the language $L(G)$ and a lexical category sequence κ' for w' such that κ and κ' are related—through the relation of similarity used in the definition of the function f?*

and Parameters—which are open to variation.

Competitive Constraint Processing The parsing process involves a series of pairwise competition between candidates. This competitive process, called *evaluation*, relies on the constraint ranking in order to determine which of the candidate structures for a given input satisfies the highest-ranking constraint. In fact, what prevails in the competition is the rank of the violated constraints: the highest ranking of the violated constraints gets the corresponding candidate structure to lose. Thus the constraints are foremost used to filter out the sub-optimal structures, and the winning candidate is the less ungrammatical one. Such a way of conceiving the parsing process makes Blache (2001)—who is interested in studying the role played by constraints in linguistic theories—present the OT constraints as *filtering* ones, as opposed to those from HPSG (Pollard and Sag, 1994), for instance, which also play a more active role in building the structure. In OT, the generation of the candidate structures is taken care of by the `GEN` function and do not involve any of the constraints. Note that another difference between HPSG and OT stands in that the HPSG constraints do not compete. For Blache, the fact that constraint failures in OT are only meaningful to the filtering process requires relying on a constraint hierarchy (constraint can be evaluated independently, but only its position with respect to the others is important), which differs significantly from traditional constraint satisfaction.

The Notion of *Structure* One problem concerns what the so-called *structures* exactly stand for. Conceptually OT is meant to cover all dimensions of language—phonology, syntax, semantics, . . . —, which justifies leaving open the type of structure, so that it suits the kind of information that needs to be encoded. Yet Blache (2001) observes that the granularity of the structure of the constrained objects directly determines the use of constraints, and impacts the type of parsing process involved. For instance, different types of parsing process are applicable whether it deals with trees or categories. If the objects are highly structured and differentiated, then a costly generate-and-test is required, whereas if the objects are more canonical, then traditional constraint satisfaction is applicable. OT, like Constraint dependency Grammar (see §2.5.1) stands in the former category, while HPSG is more—though not entirely, according to Blache—in the latter. Blache (2001) advocates the use of constraint satisfaction, arguing, among many other reasons, that the approach benefits from extensive work both theoretical and applied, to make the processing of constraints practical. Other reasons are connected to those, which make Pullum and Scholz presenting the family of MTS frameworks (see §2.3.1) as better suited to representing all aspects of language; the link between their respective arguments is the notion of constraint, which is considered as central in either case. We will come back on that aspect later.

Generation of the Candidates Another problem, related to the first one, is the lack of specification regarding the production of the candidate structures. As we said previously, the theory relies on the assumption that all the possible candidates are generated separately from the evaluation step—which handles the competition aiming to determine the optimal structure—during a preliminary step, and makes no assumption regarding what a structure should be like. The `GEN` function is responsible for the production of a possibly infinite set of candidates for the input structure, and is presented as a black box. The problem here, is that no details of `GEN` are ever supplied, especially not concerning the supply of the set of alternative candidates, which is often considered as obvious.

The Role of Constraint Violation One more issue comes from the key role granted in the evaluation process to the violation of constraints. The fact that the constraints in OT are *universal* means that they are defined independently from any context they may apply to. For instance, the constraint ranking may change from one language to another. Consequently, the constraints are often conflicting, in order to cover all aspects of language. From these conflicts inevitably arise inconsistencies, which need to be solved in order for the decision process to identify the optimal structure. Since constraint violation is necessary for the evaluation process to discriminate and pick the optimal candidate, the grammar specifies constraints which are precisely meant to fail.

Grammaticality The last source of concern about OT is with regard to grammaticality. We observe that the notion of the grammaticality of a structure is defined by reference to the one of the other candidates: a structure is defined as grammatical in OT if and only if it wins all the competitions against the other candidate, and thus turns out to be the optimal one. But is grammaticality conceptually equivalent to optimality? Is it the same thing to conclude that a structure is the best possible one for a given input with respect to a grammar, and to conclude that this input is itself grammatical? Such a conception of grammaticality does not exactly match the notion of common-sense we have discussed in §2.3.1—*i.e.* which discriminates between *expressions* and *quasi-expressions*. Let us take the problem the other way around: consider an expression Q and a quasi-expression Q', and assume a set of candidate structures for each of them (assuming that the `GEN` function is capable of providing a set of structures for both—we will come back on that assumption shortly); the optimal structure for Q is said to be grammatical in the OT sense of it, and the optimal structure for Q' is said to be grammatical as well, again in the OT sense of it. The situation thus raises the following matters:

- How to discriminate expressions from quasi-expressions?
- How to compare Q to Q' and their respective grammaticality?

These two questions remain unsolvable in native OT[6].

Keller (2000, p. 243) asserts that

> [i]n line with all major linguistic frameworks, Standard Optimality Theory (. . .) assumes a binary notion of grammaticality: the competition between candidate structures selects one candidate (or a set of candidates sharing the same constraint profile) as optimal and, hence, grammatical. All losing candidates, *i.e.*, those structures that are *suboptimal*, are assumed to be ungrammatical; Starndard OT makes no predicitions the relative ungrammaticality of suboptimal candidates.

[6] It was pointed out to us by Gerald Penn (examiner, *rapporteur*) that "hard" constraints were proposed to solve that problem, and we can only imagine that other mechanisms, such as prioritising the constraints, could be helpful as well. It is nevertheless unclear, given the very nature of OT and the role granted to constraint violation, how extensions could avoid being grammar-dependent and thus limited in scope. The role played by `GEN` also remains highly unclear, as we discuss it later in this very section.

We agree with Keller about the lack of status provided for the suboptimal structures, but we disagree with the alleged similarity between the "binary" grammaticality in OT and the binary grammaticality "in all major linguistic frameworks". The optimality-theoretic grammaticality is indeed *binary* in that it involves, by definition, no degree of grammaticality—the winning candidate is considered as being fully grammatical. However, and still assuming that `GEN` is capable of generating structures for both expressions and quasi-expressions, this notion of grammaticality strongly differs from the one in all major frameworks (we take them to be GES frameworks) in that, as we have just shown, OT considers as grammatical *any* input for which an optimal structure can be found, and regardless of whether the input is an expression or a quasi-expression. In doing so, OT loosens up the boundaries of the set of grammatical utterances in order to include quasi-expressions, and subsequently makes no differentiation between the two. Therefore the framework as such, unlike other frameworks where binary grammaticality applies, is not capable of discriminating between grammaticality and ungrammaticality—*i.e.*, any input for which candidate structures are generated necessarily ends up being optimality-theoretically grammatical.

At this stage of the discussion it is necessary to come back on the quite strong assumption we have been making all the way through: the ability of the `GEN` function to provide a set of candidate structures for both expressions and quasi-expressions. As already emphasised in §2.3.1 a generative framework is not well suited to the generation of structures for quasi-expressions; hence we are left with MTS frameworks for `GEN`. Unfortunately, to the best of our knowledge there exists no attempt to (explicitely) implement OT for syntax along with an MTS framework. Yet we can not think of any reason why it would not be possible. In fact, we will see later that the solution we propose for modelling syntactic gradience does borrow central concepts from OT, although we have not fully investigated the possibility of implementing OT with an MTS framework. The question, therefore, remains open.

2.3.3 Construction Grammar

The main and general idea governing Construction Grammar (CxG) is that the meaning of the whole is not a compositional function of the meaning of the parts, but that *constructions* carry their own meaning. The suggestion was first made by Lakoff (1977) in his paper about Linguistic Gestalts, then discussed further in Lakoff (1987b). CxG is sometimes presented as a *family* of frameworks for language studies rather than a framework as such. Early noticeable use and development of CxG frameworks are those by Goldberg (1995), Kay (1998), Kay and Fillmore (1999) among others. Other different kinds of CxG frameworks were described, such as Cognitive Grammar (Langacker, 1986, 1987), Radical Construction Grammar (Croft, 2001), Embodied Construction Grammar (Bergen and Chang, 2005), Fluid Construction Grammar (Steels and de Beule, 2006a), and others. Beyond the differences which motivate each of these frameworks, they all share the same idea that compositional semantics is not enough to for all aspects of a construction's meaning.

The Notion of *Construction* A widely accepted generalisation of the definition for the notion of *Construction* is the one provided by Goldberg (1995, p. 5) (already introduced in §2.2.3):

> C is a CONSTRUCTION *iff* C is a form-meaning pair $\langle F_i, S_i \rangle$ such that some aspect of F_i or some aspect of S_i is not strictly predictable from C's component parts or from other previously established constructions.

Such a definition implicitely makes it possible to study language phenomena such as idiomatic structures, or argument structures, as being part of plain language, whereas generative grammar and compositional semantics consider them as borderline cases. As Goldberg emphasises it,

> [t]heorists working within this theory [CxG] share an interest in characterizing the *entire* class of structures that make up language, not only the structures that are defined to be part of "core grammar". (Goldberg, 1995, p. 6)(Emphasis in original)

In CxG the form-meaning pairing is the basic unit of language. It provides a particurlarly flexible way of dealing with non-core cases. An example of argument structure is the *Caused-Motion Construction* discussed by Goldberg (1995), which was already quickly presented in §2.2.3. This case study clearly shows (i) the limitations of GES and compositional semantics, and (ii) how CxG elegantly overcomes these limitations. For more case studies within CxG frameworks see for instance Langacker (1987); Goldberg (1995); Kay (1998); Kay and Fillmore (1999).

Main Features of CxG Among the important features presented by CxG, one is that it is described by Goldberg as being "generative in the sense that it tries to account for the infinite number of expressions that are allowed by the grammar while attempting to account for the fact that an infinite number of other expressions are ruled out or disallowed". At the same time, CxG is not transformational, in that no underlying syntactic or semantic forms are posited.

Other important features are that "constructions display prototype structure and form networks of associations. Hierarchies of inheritance and semantic networks (...) are adopted for explicating our linguistic knowledge" (Goldberg, 1995). The inheritance hierarchy of constructions captures generalisations "concerning word order facts, case-marking properties, and links between semantics and grammatical relations" (Goldberg, 1995, p. 108). These are two important aspects of the theory, which make it compatible with the IG/SG conception of gradience, based on prototypicality.

Formalisation and Implementation Although a growing body of work from linguistics adopts CxG for studying different aspects of natural language, from the point of view of a computational implementation, the mathematical/logical formalisation of the framework is still rather poor. Goldberg refers to the works on Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985) and Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) as being particularly compatible with the notions developed in CxG. Noticeable formalisations and implementations are Fluid Construction Grammar (Beule and Steels, 2005; Steels, de Beule, and Neubauer, 2005; Steels and de Beule, 2006b) (FCG), and an attempt by Blache (2004) to use Property Grammars (Blache, 2001; Blache and Prost, 2005) (PG) as an operational computational framework for CxG. We will describe PG in detail in §2.5.3. The important point to emphasise here, is that the conception of language study developed in CxG shares many of its different aims and properties with Model-Theoretic Syntax (MTS). The descriptive nature of it, in particular, makes it open to describing variations and graded phenomena.

2.3.4 Preliminary Conclusions on Knowledge Representation

Although variations and graded phenomena are commonly observed by linguists at every level of natural language, from the lexical to the construtional level, their study raises various issues of knowledge representation. These problems concern the representation of the linguistic information about an input, and the representation of the gradience of natural language. In this section we have reported and discussed the literature on different types of frameworks for studying natural language, namely the families of frameworks from Generative-Enumerative Syntax (GES) and Model-Theoretic Syntax (MTS), the Construction Grammar (CxG) frameworks, and Optimality Theory (OT). For each of them we have envisaged their ability to account for graded phenomena, and how these phenomena are or could be represented.

A few preliminary conclusions can be drawn at this stage.

Firstly, the family of GES frameworks can be ruled out, as it is incompatible, by conception, with representing quasi-expressions.

Secondly, the family of MTS frameworks involves a descriptive approach to the representation of language, and a related notion of grammaticality which allow for quasi-expressions to be considered as part of natural language and for variations and graded phenomena in language to be represented.

Thirdly, Optimality Theory provides a notion of grammaticality which allows for an account of graded phenomena. Meanwhile, the variety of phenomena being covered strongly depends on the generation of a set of structures prior to applying the theory itself. The use of a GES framework for generating that initial set of structures is expected to prevent using OT with quasi-expressions.

A fourth conclusion is that the frameworks within Construction Grammar, by adopting the notion of prototype, and by their descriptive approach to language representation, present theoretical properties well suited to handle non-core grammatical features and variations in natural language.

A fifth conclusion is that the notion of grammaticality varies in every type of framework.

In GES grammaticality is binary with no possibility of variations—an utterance is either grammatical or not, and all the grammatical ones are grammatical to the same extent. The generative notion of grammaticality is probably the most standard one used in linguistic studies.

Grammaticality in MTS applies to an utterance (expression or quasi-expression) and is defined with regard to whether or not an utterance violates constraints from the grammar. MTS Grammaticality is capable of discriminating between grammatical expressions, which violate no constraints from the grammar, and ungrammatical quasi-expressions, which violate at least one constraint. Grammaticality, therefore, involves sharp boundaries around the set of grammatical expressions of a language; meanwhile it allows for quasi-expressions to be ungrammatical. Ungrammaticality in MTS is graded, in the sense that a quasi-expression can be assigned a degree according to how many rules from the grammar it violates. The model-theoretic notion of grammaticality is arguably the one capturing the best the "ordinary, common-sense notion" of it.

In OT the notion of grammaticality applies to a *structure* rather than to an *utterance*; the grammatical structure for an utterance is the optimal one. Any utterance *for which a set of candidate structures can be generated* is assigned (at least) one grammatical structure, by virtue of the fact that there exists at least one optimal structure in the set of candidates. The optimality-theoretic grammaticality does not discriminiate utterances but discriminate structures for a given input utterance. Such grammaticality accepts variations, in that no restriction is put on how many constraints are violated by a grammatical structure.

In CxG grammaticality aims to discriminate between lawful and unlawful constructions, but adopts the notion of *prototype*, which is likely—though to be confirmed—to allow for an interpretation of grammatical variations along the line of intersective and subsective gradience.

The last conclusion we can draw is that all the promising frameworks for an account of some form of gradience or another involve a descriptive approach to representing natural language, which uses a notion of *constraint* as basic unit. What this notion of constraint involves and what it precisely informs on remains to be answered.

2.4 Models of Syntactic Gradience

In this section we first review two models that specifically attempt to handle gradience. Then in the next section we look at other possible frameworks that could be used to model gradience.

2.4.1 Aarts' Model

The model of gradience devised in Aarts (2007) aims to provide a formal way to draw a conclusion when faced with cases of gradience. That is, the model is meant to be used as a framework for making a decision as to how to classify items, whose classification is ambiguous. Aarts tackles these ambiguous cases of classification from the twofold angle of Subsective and Intersective Gradience (SG and IG respectively, see §2.2). The basic suggestion is to compare the number of *characteristic properties* shown by each of the contenders.

Subsective Gradience

On the SG side, if two items belong to the same class while satisfying different amounts of characteristic properties, they are said to be in a subsective gradient relationship and are ordered by prototypicality according to how many properties they meet. The more properties are met, the more prototypical the item.

Definition 2.1 (subsective gradience, by Aarts, 2007, p. 205)

If $\alpha, \beta \in \gamma$ where γ is a form class characterized by morphosyntactic properties $\{p_1 \dots p_n\}$;

and α is characterized by $\{p_1 \dots p_x\}$, such that $0 < x \leq n$;
and β is characterized by $\{p_1 \dots p_y\}$, such that $0 < y < x$;
Then α and β are in a subsective gradient relationship, such that α is a more prototypical member of γ than β.

Intersective Gradience

On the IG side, if one item shows characteristic properties from two distinct classes, the preferred class is the one for which the item shows the most properties. The *convergence* of the item to one class or the other is simply based on how many characteristic properties are met by the item with respect to each of the two classes. Note that the model of IG under discussion is not generalised to more than two classes.

Definition 2.2 (intersective gradience, by Aarts, 2007, p. 207[7])

If $\alpha, \beta \in \gamma$ where γ is a form class characterized by morphosyntactic properties $\{a_1 \dots a_m\}$ and $\{b_1 \dots b_m\}$, respectively;
and $\exists \Psi, \Psi$ a grammatical formative which conforms to a set of syntactic properties $\{c_1 \dots c_p\}$, such that $\{c_1 \dots c_x\} \subset \{a_1 \dots a_m\}$ and $\{c_{x+1} \dots c_p\} \subset \{b_1 \dots b_n\}$;
Then α and β are in an intersective gradient relationship with respect to Ψ, and its projection ΨP.

Unfortunately, most of the language phenomena and classification issues investigated are lexical ones, even though constructional gradience is also quickly discussed.

Handling Deviant Language

The first comment that ought to be made about this model is that it is meant to stay within range of well-formedness. What Aarts is interested in is solving cases of morpho-syntactic ambiguities, and no attempt is made, apparently, to also account for deviant inputs. Therefore what we see as drawbacks or shortcomings in this model

[7] This is the original definition from Aarts. For clarity, it would have probably been preferable to make explicit that $\alpha \sim \{a_1 \dots a_m\}$ and $\beta \sim \{b_1 \dots b_m\}$.

with respect to our purpose, may not always be so with respect to Aarts' original purpose. However, it is interesting to see to what extent the proposed model may be scaled up in order to also tackle cases of deviance.

An interesting aspect of Aarts' model is precisely that gradience is captured by the linguistic properties which are *met* by an item, whereas in most other models only *violated* properties are considered. It gives, for instance, some insight on the correlation between gradience phenomena—even though within the scope of well-formedness—and the cumulativity of satisfied properties, which might turn out to be useful for designing a model of gradience. Meanwhile, considering only satisfied constraints also presents shortcomings when interested in cases of deviance.

Convergence

Another weakness, we believe, is revealed by cases of Intersective Gradience between two classes presenting a significant disproportion between the amounts of properties specifying each of them. Let us consider the following example. A and B are two classes, A being specified by 3 characteristic properties, and B being specified by 100 different properties. An item presenting two of A's properties and 3 of B's properties will converge toward B, even though 66% of A's properties are met, for only 3% of B's. For that reason we are not entirely convinced by the ability of this model to accurately capture all cases of IG[8].

Besides, the notion of *convergence* used to formalise SG and IG considers no more than two classes. That is, cases of gradience involving more than two classes are not modelled. Aarts acknowledges that it is a rather strong claim (Aarts, 2007, §8.6.4 p. 228), but it is supported by the lack of obvious evidence of phenomena involving convergence to more than two categories. Again, looking only at well-formed language we can not think of any counter-examples either. However, should we be interested in extending the model to also cover cases of deviant language, counter examples are easier to build.

[8] It was pointed out to us that there exists a body of work on how to do handle the problem correctly in the lexical semantics literature, such as so-called "overlap scores", *e.g.* Jaccard coefficients.

Relative Importance of Properties

§8.6.1 in Aarts (2007, pp. 225–227) discusses why the model does not assign any weights to properties, granting all of them the same relative importance. Thus the model considers all properties as being equally influential with respect to gradience. This choice is justified by that,

> [w]hile there are computational procedures for modelling the weighting of properties, these are not uncontroversial, for a number of reasons: first, there are different ways of assigning weightings, and there is disagreement as to which procedure is the optimal one. Secondly, weightings are based on large-scale corpora, but there is no agreement about the precise compositional make-up of such corpora. (Aarts, 2007, p. 226)

Aarts also refers to Hudson (1990) who shares the same view, to support his decision. Although it is true that there are many ways to set weights, and it is hard to know which is best, it could be argued that such a lack of an established procedure does not necessarily make the very use of weights in any way controversial. Besides, it is also likely that seeing all properties as equally influential constitutes an impediment to accurately modelling graded judgements.

2.4.2 Linear Optimality Theory (LOT) (Keller)

Keller's PhD thesis (Keller, 2000) investigates gradience in grammar, and implements a computational model for it. More specifically, his work "aim[s] (...) to investigate how constraint ranking, constraint type, and constraint interaction determine the degree of grammaticality of a given linguistic structure".

Gathered Empirical Evidences

In his thesis, Keller reports various experiments about acceptability judgement. Three main properties of gradient phenomena are identified:

> 1. [constraint ranking] constraint violations are ranked, *i.e.* they differ in seriousness. (...)

> 2. [cumulative effect] constraint violations are cumulative, *i.e.* the degree of unacceptability increases with the number of violations. (...)
> 3. [soft/hard dichotomy] two types of constraints can be distinguished experimentally: soft and hard constraints. This dichotomy captures the intuition that certain linguistic constraints are binary, while others induce gradient acceptability judgements.
>
> (Keller, 2000, p. 17)(re-formatted as a bullet-list—JPP)

Constraint Ranking Property (1) includes an implicit assumption as well, whereby only constraint violations play a role acceptability judgement, excluding de facto any role, which could be played by satisfied constraints.

Cumulative Effect Property (2) suggests that OT's account, which only relies on the one top ranked constraint to decide on grammaticality, ignoring therefore the lower ranked violated constraints, is not accurate enough.

Soft *vs.* **Hard Constraints** Property (3) introduces the discrimination between soft constraints, whose violation induces only mild unacceptability, and hard constraints, whose violation induces strong unacceptability. Intuitively, it suggests that violating a hard constraint always entails significantly more unacceptability than violating soft constraints.

Ganging-up Effect The *ganging-up effect* denotes the possibility for soft constraints to *gang up* against a hard constraint. That is, soft constraints violated by one candidate structure may sometimes join forces, in order to balance and out-rank the effect of one hard constraint violated by another—different—candidate. This constitutes an exception to the basic rule whereby the violation of a hard constraint is more unacceptable than the violation of several soft constraints.

Linear Optimality Theory (LOT)

Keller also devises a computational model for gradience, the Linear Optimality Theory (LOT), which is validated against the different properties revealed by his empirical investigation. This section presents an overview of LOT.

Keller adopts an optimality-theoretic approach to tackle the problem. See §2.3.2 for a brief overview of Optimality Theory (OT).

LOT as a Model for Gradience. LOT is designed to capture the properties of gradience, which were observed experimentally. The basic idea is to extend OT so that the sub-optimal structures are no longer ignored, but ranked in order of acceptability.

Handling Constraint Ranking The property of *Constraint ranking* is obtained by weighting the constraints, a weight being proportional to the "amount" of unacceptability it yields when violated. Thus, mild unacceptability is represented by a low value, whereas strong unacceptability is represented by a high value. Subsequently the third property, that is the *soft/hard dichotomy*, is obtained as a side-effect, in that it is derived from the interpretation of the constraint weights: a constraint with a low weight is soft, whereas a constraint with a heavy weight is hard. As we will see next, the constraint weights are deduced from the empirical values, as a result of a solving process. One of Keller's contributions is to use a machine learning algorithm in order to determine the appropriate weights to be associated with the constraints, so that the subsequent degree of acceptability matches the empirical judgements.

The Constraints' Weights. Keller correlates the empirical observations of acceptability judgement with the numeric weights by setting a system of equations. The solution to the system is the set of weights to be assigned to the constraints. Given a set of candidate structures and the degree of acceptability observed for each candidate, all the candidates are compared pairwise, which gives a system made up of $k(k-1)/2$ equations (where k is the number of candidates). The difference in acceptability between two structures must then equal the difference between the weighted sums of violations for the two structures. Solving the system consists in finding a solution, that is a tuple of numeric weights, for the system of equations.

Handling the Cumulativity Effect The degree of acceptability of a candidate structure is computed by summing up the weights of all the constraints a candidate structure violates. By doing so, the scoring function captures the *cumulativity effect*. This function is then used to rank different candidates. We notice that this function only takes the *violated* constraints into account, ignoring the satisfied ones. We will come back later to that important aspect.

Handling Soft *vs.* **Hard Constraints** As far as the soft/hard classification is concerned, unfortunately at this stage it remains rather intuitive. The modelling study ♯1, p. 283, for example, concludes that some of the constraints under investigation are *hard* by virtue of the fact that the computed weights are "high", as opposed to other constraints weights, which are "considerably lower than the ones of the hard constraints" (p. 286). The results would be nicely complemented by the identification of a decision threshold between the two soft and hard classes, since it is not merely a rough estimation of the value of their weight, but the dichotomy is also meant to model differences in behaviour (*e.g.*, context-dependent *vs.* context-independent, cross-linguistically stable *vs.* not).

Shortcomings of LOT

Satisfied *vs.* **Violated Tradeoff.** Keller (2000), in using OT, implicitly assumes that acceptability only depends on the violated constraints, the satisfied ones being of no importance, especially for the cumulativity effect. In our opinion, acceptability can not be accurately modelled by looking at the violated constraints only: there are reasons to believe that a tradeoff between the influence of satisfied and the influence of violated constraints is also relevant. Although experimental evidence are presented to support such the approach, we could not find any convincing argument in the reported experiments with respect to that hypothetical tradeoff. Experiments are indeed carried out in order to test different sorts of constraint interactions (see Chapters 3, and 4), but only constraint violations are considered; the interaction between violations and satisfactions is never tested.

Different reasons can be presented to support the idea of a tradeoff between satisfied and violated constraints. One reason could be summarised as "positive ganging up effect", in reference to Keller's "ganging up effect". In the positive version of it, one could argue that many satisfied constraints might gang up against a significantly smaller number of violated ones. The satisfied constraints, when in sufficient number, would weaken the negative strength of the violated ones. This is obviously only a hypothesis that ought to be confirmed empirically. But in any case, it seems reasonable to consider the possibility and therefore to expect a computational model for gradience to account for it.

Another reason is related to the discrimination between soft and hard constraints—recall that the discrimination is a matter of inducing either mild unacceptability (soft constraints) or strong unacceptability (hard constraints) when violated. Let us, for example, consider two distinct input strings, S_1 and S_2.

- S_1 is such that:
  - one hard constraint is violated (we note $P^-(S_1) = \{1H\}$), and
  - one hard constraint is satisfied (we note $P^+(S_1) = \{1H\}$).
- S_2 is such that:
  - exactly the same one hard constraint is violated as for S_1 (*i.e.*, $P^-(S_2) = P^-(S_1) = \{1H\}$), and
  - one soft constraint is satisfied (we note $P^+(S_2) = \{1s\}$).

It seems reasonable, in this case, to hypothesise that the acceptability of S_2 may be lower than the one of S_1; not because of the violated constraint, which is exactly the same in both cases, but because of the satisfied constraint in each case (*i.e.* $P^+(S_1)$ and $P^+(S_2)$). OT, and Keller's experiments in particular, fail to discriminate between S_1's acceptability and S_2's acceptability, since only *violated* constraints are accounted for.

Grammaticality *vs.* **Optimality** As we previously emphasised it in §2.3.2, the notion of grammaticality in OT differs from those in generative grammar and in MTS. Keller augments the optimality-theoretic notion of grammaticality in introducing degrees according to how many constraints a structure violates. Similarly, Hayes (2000) suggests to complement OT with *bands of values* in order to introduce a continuum of well-formedness. However, in either case grammaticality is still inherently optimality-theoretic: it applies to a structure rather than to an input, and therefore it only allows comparisons between grammatical inputs—*i.e.*, between inputs for which a set of candidate structures was generated. Ungrammaticality, and furthermore degrees of ungrammaticality, are not accounted for.

Implementation As far as implementing a model of gradience for natural language processing is concerned, experiments with Keller's model were only done using sets of candidate structures which were generated manually. A machine learning algorithm was proposed in order to automatically compute constraint weights from empirical data, but the origin of the set of *inputs* results from a series a assumptions, for which no algorithm or reference to algorithms is provided. Although, as emphasised by Keller, keeping his "assumptions regarding the input as minimal and as theory-neutral as possible" allows him "to make claims of maximal generality", it prevents us from drawing definite conclusions regarding the computational adequacy of the full model.

2.5 Implementing Syntactic Gradience

In this section we look at possible frameworks that could be used to model gradience. In particular, we look at approaches related to Constraint Dependency Grammars, Property Grammars and Configurations.

2.5.1 Constraint Dependency Grammar (CDG) (Maruyama)

In Constraint Dependency Grammar (CDG) the grammar is made up of a set of statements about how words are connected together by dependency relations. These statements are constraints, which correspond to the set of conditions of well-formedness to be met by a string from the language. A constraint takes the form of a logical formula. The variables involved are *words* and their different possible *roles*. According to Maruyama (1990):

> (...) each word i in a sentence s has k-different roles $r_1(i), r_2(i), \ldots, r_k(i)$. Roles are like variables, and each role can have a pair $\langle a, d\rangle$ as its value, where the *label* a is a member of a finite set $L = \{a_1, a_2, \ldots, a_l\}$ and the *modifiee* d is either $1 \leq d \leq n$ or a special symbol `nil`. An analysis of the sentence s is obtained by assigning appropriate values to the $n \times k$ roles (...).

In other words, assigning the pair $\langle a, d\rangle$ to the role $r_1(i)$, for instance, means that the role r_1 is to be in the a dependency relation with the modifiee d. An individual parse tree for s is a solution, which satisfies simultaneously all the constraints in the grammar.

A range of works are based on, or extend Maruyama's Constraint Dependency Grammar (CDG); we will go through some of them later in this chapter.

For Maruyama (1990, p. 32):

> a constraint C is a logical formula in a form
>
> $$\forall x_1 x_2 \ldots x_p : \text{role}; P_1 \& P_2 \& \ldots \& P_m$$
>
> where the variables $x_1, X_2, \ldots x_p$ range over the set of roles (...).

Examples of constraints[9] are:

$$(G1-1) \quad word(pos(x)) = D \Rightarrow (lab(x) = \text{DET}, \\ word(mod(x)) = N, pos(x) < mod(x))$$

"A determiner (D) modifies a noun (N) on the right with the label DET."

$$(G1-2) \quad word(pos(x)) = N \Rightarrow (lab(x) = \text{SUBJ}, \\ word(mod(x)) = V, pos(x) < mod(x))$$

"A noun modifies a verb (V) on the right with the label SUBJ."

These constraints are *topological* ones, in that they describe what the topology of the output structure should be like. Each of these constraints describes a dependency,

[9] The constraints' labels are those from Maruyama (1990)

that is an arc in the output structure. In a constraint, an implication of the general form $x \Rightarrow y$ can be read "x modifies y", and represents a dependency relation between x and y.

The Parsing Process In basic CDG the constraints are not all involved at the same level; two classes of constraints can be identified, according to the role they play in the parsing process. Parsing with CDG happens in two successive phases, hence the two classes of constraints.

The first phase only involves a subset of the constraints, which constitute the *"core" grammar*[10]. This first phase aims to build a *constraint network* for a specific input sentence. The resulting network only involves the constraints present in the core grammar. The network can be seen as a compact representation of all possible parses for the core grammar. The second phase is concerned with pruning the network. This is achieved by constraint propagation.

Constraint propagation refers to mechanisms of modifications of the constraint system by means of inference rules. These rules aim to maintain/enforce local consistency for subsets of variables. Generally speaking, and to exemplify the process[11], let us consider the two variables x, y, whose domains are respectively [23..100] and [1..33], and a constraint $x < y$. Constraint propagation can narrow down the domains to $x \in [23..32]$ and $y \in [24..33]$.

Returning to Maruyama, the constraint propagation mechanisms are triggered by adding new constraints to the system. An example of such a constraint is the following (where fe(i) is a function provided for accessing features): $(G2b-1) \quad word(pos(x)) = PP, \texttt{on\_table} \in \mathrm{fe}(pos(x)) \Rightarrow \neg(\texttt{floor} \in \mathrm{fe}(mod(x)))$

"A floor is not on a table." Following their introduction the constraints are tested against each value and pair of values in the network, and the domains are subsequently modified to maintain consistency.

[10] Maruyama makes no reference whatsoever to the core grammar from Principles and Parameters, and actually puts *core* into double-quotes.

[11] Example borrowed to Christian Schulte and Gert Smolka, from the web page at `http://www.mozart-oz.org/documentation/fdt/node4.html` as it stands in October 2006

2.5.2 Weighted Constraint Dependency Grammar (WCDG) (Schröder et al.)

The Weighted Constraint Dependency Grammar (WCDG) presented in Schröder (2002) and also discussed in KilianFoth, Menzel, and Schröder (2000); Foth (2004); Foth, Menzel, and Schröder (2005); Menzel (1998); Schröder et al. (2000) among others, is defined on top of Maruyama's Constraint Dependency Grammar.

In his original work, Maruyama is only interested in parsing well-formed input. Menzel (1995) extends CDG in different ways, in order to deal with robustness in language processing (see, for example, Foth, Menzel, and Schröder (2005) for an overview of robust parsing). Successive developments of this work (Menzel, 1998; Foth, Schröder, and Menzel, 2000; Foth, Menzel, and Schröder, 2005) led to the formal definition of the Weighted Constraint Dependency Grammar formalism (WCDG) (Schröder, 2002).

The formalism is of interest to us for different reasons: it acknowledges and formally accounts for a fuzzy notion of grammaticality; it is a constraint-oriented formalism; it relies on a scoring function, which is applied to an input utterance; it accounts for a fine-grained discrimination between soft and hard constraints along a continuum, by weighting them. We present here an overview of the formalism.

Unlike more traditional strategies for achieving robustness, which rely on *post mortem* processing modules in order to recover from an error, with WCDG robustness is achieved "naturally" in that it derives from the basic parsing mechanism. Among the different aspects usually tackled by a robust processor, the focus here is essentially put on dealing with ill-formed input.

Multi-layers Extension

Menzel advocates an independent account of different dimensions of language (*e.g.* syntax, semantics, pragmatics, ...), as opposed to an integrated one. Examples of integrated accounts are those in HPSG or CxG, where syntactic, semantic, and sometimes also pragmatic components are combined within the same structure—a feature structure for HPSG, and a Construction for CxG. Menzel extends CDG (initially restricted to purely syntactic constraints) in introducing *representation layers*. Menzel (1995) only discusses two layers: a syntactic and a semantic one, but the idea was extended later to other dimensions of language. With Schröder (2002) the notion of layer is then generalised further and other levels, such as the FUNCTOR-ARGUMENT level are introduced.

The eXtended Dependency Grammar (Duchier, 1999; Debusmann, Duchier, and Kruijff, 2004) (see §2.5.8), also based on CDG, adopts the same kind of multi-layer approach and formalises it using multigraphs.

Constraint Weights

The weights in Schröder (2002) intuitively capture how important a constraint is, and are established on the basis of the frequencies observed for a particular phenomenon in a corpus—though they could also be arbitrarily set by the grammar writer. The extracted frequencies are then used to determine penalties to be assigned to the constraints involved in modelling the studied phenomena, so that the combination of the weighted constraints leads to recognising these phenomena with the same frequency as the one observed on corpus. Note that there are not supposed to be any ill-formed inputs in the corpus, but our understanding, very informally, is that these frequencies concern fine-grained phenomena, which are then mapped to the constraints involved in parsing them in order to figure weights to be assigned to these constraints.

Dynamic Weights The *dynamic weights*, in Schröder (2002), are a nice mechanism, which makes it possible for weights to be functions of different dynamic features. This way, the influence of a constraint on the overall score of the structure can be, to some extent, controlled depending on the type of construction involved. The constraint such a dynamic weight is assigned to is called a *dynamic constraint*.

In §5.8 of Schröder's thesis, the mechanism is illustrated with a case of extraposition of a relative clause. The underlying problem being discussed has to do with ways of dealing with inconsistencies in the grammar. The problem is exemplified by Schröder with the case of two different constructions, which must be captured by inconsistent constraints. The example is a case of relative clause, with the relative pronoun being either extraposed or not. The way the WCDG grammar captures the contradiction between the two constructions is by having two contradicting constraints, each of them being assigned a specific weight.

The source of inconsistency stands in that on one hand one constraint specifies that the two anchors of the dependency must stick next to each other, and on the other hand another constraint specifies that the two anchors of the dependency must be distant from each other from a certain distance. Quite obviously a construction can not satisfy both at the same time: in the case of German, either the pronoun is extraposed, or it is not. Thus in either case, one of the two constraints is necessarily violated. It is then necessary to make a choice between the two structures: which of the two constraints should prevail, when faced with one of these constructions? The answer provided in Schröder (2002) relies on the use of so called *dynamic weights*, assigned to the constraints involved in the inconsistency. In the example exposed by the author, this dynamic weight is a function of the distance between the two anchors of the dependency, since it has been identified as a key factor of choice.

Unfortunately, the author does not discuss the correlation between the linguistic aspect, which is captured by the constraint itself, and the feature captured by the dynamic weight. Thus at the end it is not clear what is precisely captured by the weighted constraint.

To summarise, dynamic weights make it possible to penalise a constraint according to how frequent a construction is.

Gradation and Acceptability Judgement Schröder argues that *gradation* exists in natural language at different levels, and focuses on *linguistic preference*, *grammatical acceptability*, and *uncertainty*. Schröder observes (§§5.8.1 to 5.8.3) on a corpus a statistical correlation between the word count of a relative clause, and the extraposition or not of the relative pronoun, then draws a conclusion about the acceptability (we take it here to refer to gradation as a whole) of extraposed relative clauses:

> These corpus statistics suggest that both length and distance influence the acceptability of extraposed relative clauses. However, the latter seems to be a much stronger indicator. (Schröder, 2002, end of §5.8.3, p.178)

While we agree on that statistical distributions provide a good account of linguistic preference and uncertainty, we are still not convinced that these are suitable, by themselves, to grammatical acceptability. The rarity of a sentence or a particular construction does not make it necessarily less grammatically acceptable. The relationship assumed by the author between the statistical distribution of extraposed relative clauses in a corpus, and their gradation seems more to cover linguistic preference and uncertainty than to cover grammatical acceptability. The choice of values for dynamic weights[12] is later justified as follows:

> Since the acceptability of an extraposition changes gradually depending on the length of the relative clause and the extraposition distance, dynamic constraints (...) whose weight depends on the actual construction should be employed. (Schröder, 2002, p.179)

As far as we understand it, statistical distributions only are taken into account for setting these weights; we wish that some form of human judgement had been taken into account as well (at least in the evaluation) in order for gradation to account for the grammatical acceptability of an utterance, as it is claimed it does.

The weights in WCDG are meant to privilege the most frequent constructions against the less frequent ones. In the context of choosing which structure best fits a construction, it is indeed meaningful to give priority to the most frequent structure. It is also meaningful to use such weights in order to model linguistic preference or to resolve ambiguities among different readings of the same input, all readings being equally acceptable. However, we find these penalties to be quite different from those also based on statistical distributions and used, for instance in Keller, in order to capture the role of constraints with regard to a construction's grammatical acceptability.

Scoring Function

A preference mechanism among the constraints is then introduced by weighting them according to their importance. A solution violating constraints is no longer ruled out,

[12] Note that the term *dynamic constraint* used by Schröder refers to a static constraint whose weight is dynamically computed according to context, rather than to a constraint which would have some dynamic properties.

but assigned and ranked according to a *confidence score*. The *confidence score* results from multiplicatively combining together the weights of the violated constraints. A weight is a *penalty factor*, which is meant to weaken the constraint by influencing negatively the *confidence score* associated to a solution of the constraint satisfaction problem.

Then the role of the *selection procedure* is to deal with the structural ambiguities. For Menzel (1995, pp. 29–30),

> the selection procedure is based on a local assessment function heuristically identifying relations to be pruned.

Menzel simply refers later to a sum of quadratic errors as the assessment function being used for experimental purposes.

This function is meant to model the acceptability of the dependency structure.

Preference-based Reasoning

The notion referred to by Menzel as *preference-based reasoning* constitutes the backbone of the eliminative parsing strategy adopted to prune the search space of candidate solutions. Bearing in mind that the underlying techniques are borrowed to the field of constraint satisfaction—which includes constraint propagation—, the basic reasoning is "(...) complemented by a second propagation principle based on preference-induced constraints. These are activated only in situations where enough positive evidence can be derived from almost uniquely determined preferences." (Menzel, 1995, p. 31).

These *preference-induced constraints* can be seen as inference rules, where the premises are the expression of a preference, which, when met, activates a new constraint. This newly-activated constraint is then propagated through the constraint network and required in turn to hold for the input configuration. The advantage of such a mechanism is that it prunes further the search space by constraint propagation, but only for a preferred configuration.

2.5.3 Property Grammars (Blache)

We only give here a brief introduction to Property Grammars, in order to discuss a few implementations of it. We go into thorough details in ch. 3.

Property Grammars (PG) was initially defined in Blache (2001) on the basis of the 5P formalism (Bès and Blache, 1999; Bès, Blache, and Hagège, 1999). Various developments of it were then published (VanRullen, Guénot, and Bellengier, 2003; Blache, 2005), including sometimes variants in some definitions. PG is a paradigm for representing natural language, which aims to provide a framework suitable to adopt a descriptive approach to language. By descriptive, we mean that the grammar is made up of a collection of statements about language (called *properties*), in the sense of MTS (see §2.3.1). A *property*, in the sense of PG, is a *constraint*, which models a relationship among syntactic categories. A constraint is of one of the pre-defined types, and takes the form of an n-ary predicate. For example, a property of Linearity between a Determiner (DET) and a Noun (N) (noted $Linearity(DET, N)$ or $DET \prec N$) holds true iff the determiner precedes the noun. Given the input utterance “the red book” this Linearity constraint thus holds true between 'the' and 'book'. A grammar is a constraint system made of all the properties of the language. Thus, parsing an input sentence according to a PG grammar yields a collection of statements about the input, which either hold or fail. Such a collection of statements—whether they hold or fail—characterises the input and is called *characterisation*.

The grammar also specifies syntactic constructions. The notion of construction used in PG is similar to the one in Construction Grammar (CxG). It ranges from a lexical item's Part-of-Speech, to phrasal structures and top-level constructions such as, for example, the Caused-motion or the Subject-verb Inversion constructions. Each construction is specified by a feature structure and by a subset of properties from the grammar. The feature structure allows specifying more fine-grained properties using feature unification, *e.g.* in: $Linearity(\mathrm{DET}_{[person,gender]}, N_{[person,gender]})$ An example of toy grammar is presented in Chapter 3, §3.2.4. An interesting aspect of PG is that it is not tight to building a dependency structure, as in CDG. In fact, the constraint system may contain any type of relation: *e.g.* dependency, constituency, uniqueness, etc. but also user-defined relations. The formalism is thus very flexible when it comes to write the grammar.

Pros and Cons

From the perspective of modelling gradience, PG presents essentially two important pros. One is with respect to the amount of syntactic information that can be represented: the characterisation of an utterance informs not only on the constituent structure and the feature structure, but also on the syntactic relationships among constituents, whether met or broken. It is an elegant way of keeping track of all the different properties among constituents, which are otherwise not contained in either of the tree structure or the feature structure. Such a representation of syntax makes PG an MTS framework and is, therefore, especially well-suited to representing and reasoning about both expressions and quasi-expressions.

A second one is with respect to the suitability of the characterisation to addressing problems of gradience: the similarity that it presents with the *syntactic charateristics* of an item in Aarts' model of gradience is striking. Just as well as Aarts' characteristics serve as features in the categorisation of an item and for addressing the related questions of gradience, we will see later, and in many occasions, in this dissertation that PG's characterisation of an utterance can do exactly the same and even more.

Besides these two main aspects, we can also stress that a property in PG may apply to different dimensions of language—syntax, semantics, pragmatics, . . . We already saw in §2.2.3 an example of how such a feature can turn out to be quite useful for conditionning syntactic categories partly on semantic properties, for instance.
Another interesting feature is that the paradigm seems not to be tight to a specific type of analytical linguistic structure, such as a dependency structure, or a phrase structure. A word of caution is however necessary, since it has only been implemented so far for phrasal constituent structures.

As for cons, the main one is the exponential processing cost of a constraint-based approach. Yet, various heuristics can be implemented in order prune the search space and drop the complexity to a more practical level.

2.5.4 SeedParser (VanRullen)

The SeedParser is a syntactic parser for Natural Language, presented in VanRullen (2005). It is not a model of gradience as such, but it addresses, more or less explicitly, many of the different issues we are dealing with. Most importantly it accounts for degrees of grammaticality. Therefore we find it quite relevant to present here.

Another reason for putting the SeedParser under scrutiny is that it implements the Property Grammars framework, which is the one we have also chosen to use. We will come back later on the motivation for such a choice in our case.

Scoring Function

The SeedParser from VanRullen (VanRullen, 2005) uses the *density of satisfied constraints*[13] in order to guide the parsing process by sorting the partial parses. Informally the notion of *density* is the proportion of satisfied constraints, and is defined as the ratio between the number of satisfied constraints and the total number of constraints being evaluated. Priority is given to the partial parses with best densities, when building the upper levels. An advantage of VanRullen's density is that, unlike with Menzel's and Keller's functions, the satisfied constraints are now also taken into account. Even further, some degree of interaction between satisfied and violated constraints is accounted for by the ratio satisfied/violated. However, as discussed in Blache and Prost (2005), such a "naive" ratio is not sufficient to capture many important features involved in gradience.

Let us consider the following examples:

(2.3) (a) *Quelles histoires Paul a-t-il écrites ?*
What stories Paul did-he write[fem-plu]?
'What stories did Paul write?'

(b) *Quelles histoires Paul a-t-il écrit ?*
What stories Paul did-he write[masc-sing]?
'What stories did Paul write?'

[13] note that the term *density* is used both in VanRullen (2005) and Blache and Prost (2005) but it corresponds to a different formulation in each case. The difference is irrelevant at this stage, and is detailed in Chapter 5.

(c) *Quelles histoires a-t-il écrites Paul ?*
What stories did-he write[fem-plu] Paul?
'What stories did he write Paul?'

(d) *Quelles histoires a-t-il Paul écrites ?*
What stories did-he Paul write[fem-plu]?
'What stories did he Paul write?'

These examples are given in order of (un)acceptability which corresponds in our hypothesis to a progressively greater number of violated constraints. Constraints are given here without taking into account specificities of the interrogative construction:

(C9) $\mathrm{NP[obj]} \rightsquigarrow \mathrm{VP[ppas]}$
(C10) $\mathrm{NP[subj]} \prec \mathrm{VP}$
(C11) $\mathrm{NP[subj]} \prec \mathrm{VP},\ \mathrm{V} \nLeftrightarrow \mathrm{NP[subj]}$

Even without a precise evaluation of the consequence of constraint violations type by type, this first criterion can constitute an objective element of estimation for acceptability: unacceptability increases with the number of constraint violations (This observation corresponds to Keller's property of *Cumulativity*). This indication seems trivial, but directly comes from the possibility of representing separately the different types of syntactic information by means of properties. Such estimation is for example not possible with a phrase-structure representation and even difficult using classical constraint-based approaches such as HPSG.

However, it is necessary to have a finer-grained use of such information. In particular, the number of constraints may vary from one category to another. Some categories, such as Adverbial Phrases, are very static and are described with a limited number of properties. At the other end of the spectrum, the Noun Phrase, which can have many different forms, needs an important number of properties. It is then necessary to distinguish the number of constraint violation in these cases: violating a constraint for an AdvP entails more consequences regarding acceptability than for the NP. Again, this indication is purely quantitative and does not take into account constraint type. It is probably the case that some constraints (*e.g.* exclusion) play a more important role with respect to acceptability than dependency for example. The notion of density used by VanRullen does not account for such differences among categories.

Grammar Graph Design

In VanRullen (2005) the SeedParser relies on a rather complex multi-graph structure, where all the information manipulated during the parsing process is stored. One aspect of this representation consists in connecting a type of category (*i.e.* construction) to every property defined in the grammar. For example, the Noun Phrase construction, which is encoded as a node, is connected to every property which involves an NP in its definition (*e.g.* $\mathit{Linearity}(VP, NP)$, or $\mathit{Exclusion}(NP, VP)$). Moreover, every property is also connected to all the constructions it belongs to in the grammar (*e.g.* a $\mathit{Linearity}(DET, N)$ is part of the definition of an NP construction, thus is connected to it). That is, from any one construction one can access all the properties it is involved in, and vice-versa from any property one can access directly all the constructions for which the property is involved in the definition. Say, for example, that the NP construction is defined by—among others—a property $\mathcal{P}_1 = \mathit{Linearity}(DET, N)$, and a property $\mathcal{P}_2 = \mathit{Uniqueness}(N)$, then the NP construction is accessible from both the $\mathcal{P}_1$ and $\mathcal{P}_2$ nodes; $\mathcal{P}_1$ is accessible from DET and from N, and $\mathcal{P}_2$ is accessible from N.

One positive aspect of this representation is that it makes it possible and easy to fetch all the relevant constraints that need to be taken into account given one single construction. It also makes it easy to find which constructions may be triggered given a property. Such a feature is useful during the parsing process first when it comes to listing all the properties which hold on the categories/POS for a given input, then again when it comes to decide which are the categories that can be inferred given a set of properties. However, as pointed out in VanRullen (2005) many attempted tests will fail—even though their failure was predictable, thus impacting the process's performance.

Parsing and Constraint Solving Strategies

The SeedParser adopts an incremental, left-to-right, bottom-up parsing strategy. As an incremental strategy, it builds a new node by progressively incrementing the set of its children with new constituents. There is a major inconvenient in such an approach.

Every time a set of constituents is incremented with new ones the corresponding constraint system must be evaluated again with the new constituents. Such a revision step is required because of the non-monotonicity of some of the constraint types. Informally, monotonic constraints in PG are those whose truth values do not change when the set of underlying independent variables grows by addition of new ones; conversely, non-monotonic PG constraints are those whose values are not stable when the set of variables grows. For instance, a constraint of uniqueness of a constituent type which holds for a particular set of constituents may fails when the set is incremented with a new instance of constituent from the same type as the one the constraint is specified for. The constraint must, therefore, be revised when the set of constituents is incremented.

The consequences of the revision process are twofold. One concerns the high processing cost associated with it. Unfortunately VanRullen does not thoroughly detail the implementation of the constraint satisfaction process in itself, saying simply that it involves evaluating the constraint system while walking through the graph structure. Besides the fact that the phase of constraint evaluation is admittedly not optimised, and that some linguistic phenomena were intentionally ignored in order for the SeedParser to output a result within “reasonable” time, we see as a main inconvenient the necessity of having redundant evaluations of the same constraints. It is inconvenient precisely because redundant, and because of the subsequent complexity of the process.

The second inconvenient, prior to be detailed, requires a digression about the bottom-up aspect of the strategy. It involves the *projection* step, which can be seen as a top-down prediction, whereby the category of a new node is decided. This category results from an inference mechanism, which takes the satisfaction values of the constraints in the system as premises. Without going into the detail of the a mechanism, the fact that it relies on the satisfaction values means that it is directly impacted by the revision of the constraint system. If the satisfaction values change, then the inferred categories must be revised too. In theory, the categories already inferred might no longer be inferrable, and new categories might become inferrable. It seems that the SeedParser handles the latter case, but we could not be entirely convinced that the former case is properly handled as well. Categories, therefore, might not be considered for projection as mother node.

If part of the search space is ignored as it is, then one important claim made in PG whereby the parsing process is not necessarily head-driven, no longer holds. The claim is indeed based on that a category can be projected by any type of constraints, unlike in head-driven frameworks where the projection relies exclusively on head constraints. It is, indeed, an important feature from PG, which allows the development of parsing strategies highly robust: if it is enough for a set of set of constituents to meet *any* constraint in order to be able to infer a mother for them, then it opens up the number of possibilities, and thereby the search space, for parsing ill-formed language since finding a head constituent is no longer required.

In ignoring part of such an open search space the SeedParser reduces accordingly its capacity to be robust. It is an impediment to robustness which we overcome with the parsing strategy we develop in Chapter 4. This being said, we agree with VanRullen that exploring the entire search space is obviously extremely costly, and especially so in the context of an incremental strategy, where the revision process is already quite costly. That is why VanRullen explicitely chooses to ignore, during the projection phase, the constraints presenting the property of *lacunarity*, which corresponds to a property of non-monotonicity. However, we think that a better option for pruning the search space consists of:

- adopting a non-incremental strategy, without ignoring any type of constraints, but

- adopting also a numerical heuristic.

It is such a solution that we develop in Chapter 4.

2.5.5 Dahl and Blache

Dahl and Blache (2004) presents an incremental parsing strategy for Property Grammars, which is implemented in CHRG (Constraint Handling Rules Grammar, (Christiansen, 2005)). CHRG is a meta-language for the definition of grammars in the constraint programming language Constraint Handling Rules (CHR) (Fruehwirth, 1998).

As Dahl and Blache rightfully put it, "[CHRGs] are to CHR what DCGs[14] are to Prolog. Just as DCG rules complie into Prolog rules, CHRG rules compile into CHR rules". The resulting parser can be described as using an incremental, left-to-right, bottom-up strategy, with top-down prediction.

The strategy is a bottom-up one because it builds a parse tree starting from the terminals (a sequence of lexical categories), up to the root. The licensing process, which infers new non-terminal constituents, proceeds incrementally by grouping together contiguous constituents, from left to right. Such a group of constituents constitutes an assignment for a subset of variables involved in the CSP. A top-down prediction is made with the first two constituents being grouped, which yields the category of the new non-terminal on the upper level. New contiguous constituents are then tested against the predicted category (*i.e.* the mother constituent), as long as a relationship of Constituency can be established between mother and this new daughter. Let us consider, for example, the sentence:

(2.4) The red book burns

Grouping the determiner (DET) in position ♯1 with the adjective (ADJ) in position ♯2 yields to predict the Noun Phrase (NP) category for the new non-terminal mother constituent. Attempts are then made to incrementally add new contiguous daughters, provided they meet constituency requirements from the mother. In this case the noun (N) in position ♯3 is successfully added to the sequence of daughters, but the attempt to add the verb (V) in position ♯4 fails, since a verb does not meet the constituency requirements of an NP.

As part of the incremental process, with every new assignment—or more precisely with every newly augmented assignment, comes a new subset of constraints from the CSP, whose satisfiability needs to be assessed. Following up with the last example, the process starts with the assignment $\mathcal{A}_1 = [DET_1, ADJ_2]$. The subset of constraints concerned with variables taking values in $\mathcal{A}_1$, called "subset of *relevant* constraints" with respect to $\mathcal{A}_1$, is identified, and its satisfiability is checked.

Following is the top-down prediction step, which relies on *inheritance* mechanisms

[14] DCGs: Definite Clause Grammars

among properties. These mechanisms raise issues, which are addressed later in this thesis.

2.5.6 Morawietz and Blache

Morawietz and Blache (2002) describe a CHR implementation of a parser for PG. It constitutes the first parser for a constituency-based approach, which exclusively relies on constraint programming techniques. The proposed implementation simply aims to show that, unlike with other constraint-based formalism, a parser for PG can rely on constraint programming techniques only.

The general strategy adopted is (again) incremental, left-to-right, bottom-up, with top-down prediction. Besides being a "toy implementation", merely serving the authors' point as a proof-of-concept, this parser presents the shortcomings we have already discussed about this type of approach.

2.5.7 Configuration Task

Before we look into more frameworks we need, for upcoming references, to introduce what is involved in solving a problem of configuration. Definition 2.3 reproduces the definition of a general *configuration task*, as provided in Mittal and Frayman (1989, pp. 1395–1396).

Definition 2.3 (Configuration Task, Mittal and Frayman, 1989)

> *Configuration is a special type of design activity, with the key feature that the artifact being designed is assembled from a set of pre-defined components that can only be connected together in certain ways. (...) [W]e define a configuration task as follows:*
>
> **Given:** *(A) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at the port, and other structural constraints (B) some description of the desired configuration; and (C) possibly some criteria for making optimal selections.*

> **Build:** *One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set, or, detect inconsistencies in the requirements.*

Component-based Architecture

component Mittal and Frayman discuss a component-based (C-B) architecture in order to address the configuration task from a general prospective. A *component* is described by:

- a set of properties,
- ports for connecting it to other components,
- constraints at each port, and other structural constraints.

. Note that a component is composite structure, and a sub-component relationship must be explicitely specified.

Ports The notion of *port*, as described by Mittal and Frayman, is the access point to the "outside-world"; in a generic C-B model, it represents the location where other components may be connected. For example, if we take the example of a computer, the motherboard will have different ports for all sorts of extension cards to be plugged in.

Conceptually, if we see a constituent structure as a component, the ports would implement the connectivity of this component. That is, they could be used to encode relationships of Immediate Dominance and Linear Precedence among components.

The Restricted Configuration Task

As shown in Mittal and Frayman (1989), the general task of configuration involves exponential search in the worst case, the search space of all possible configurations being on the order of $\sqrt{(pN)!}$, with N the number of components, and p the number of ports per components. However, the worst case can be improved by introducing different restrictions on the general task, which helps reducing the complexity. Different approaches are thus discussed in the literature according to the type of restrictions adopted.

We only report here a few of the possible restrictions.

Typical Restrictions

Functional Architecture Mittal and Frayman (1989) adopt a *functional architecture*, which is based on the observation that "[a]rtifacts are typically (but not always) designed with some purpose in mind". The components are considered to exercise functions, and the specifications of an artefact include functional requirements, which must be met by the components. In the context of natural language processing, the functions could play the role of grammatical functions, such as *subject*, *verb*, *direct object*, etc.

Key Component Another restriction presented in Mittal and Frayman (1989) relies on the notion of *key component*, which is pre-identified as being "crucial to implementing some function". Should we be interested in building a head-phrase structure such a restriction would typically concern heads.

Specified Target An important restriction often adopted is to take into consideration the specification of the desired product. In such a case, the number of configurations to be considered is reduced by aiming at only one specific product. In Sabin and Freuder (1996), for example,

> [t]he specifications for an actual product describe the requirements that must be satisfied by the product.

And subsequently,

> [t]he solution has to produce the list of selected components and, as important, the structure and topology of the product.

2.5.8 eXtended Dependency Grammar (XDG) (Duchier)

Duchier (1999) presents an axiomatisation of the Dependency Grammar syntax, and formalise parsing as a configuration problem. Further developments of the basic model (Duchier, 2001, 2003; Debusmann, Duchier, and Kuhlmann, 2005) led to the definition of the grammar formalism called eXtensible Dependency Grammar (XDG) (Debusmann, Duchier, and Kruijff, 2004).

The main argument put forward in favour of such a formalisation (and formulation) is to benefit from Concurrent Constraint Programming technologies, which are quite successful at solving configuration problems. In this model, "an elegant and concise axiomatic specification of syntactic well-formedness becomes naturally an efficient program [for parsing]" (Duchier, 1999, p. 115). The well-formedness conditions are thus global, and parsing is a process of model enumeration, then model elimination. These well-formedness conditions, that is, these axioms, are implemented as constraints over nodes and labelled edges between nodes.

Different *principles* are applied to axiomatise the notion of well-formed dependency tree. The principles are implemented as a specific type of constraints in the grammar. They hold either on a single or on multiple dimensions. These principles are concerned, for instance, with a node's valency, an edge's label (*i.e.* a *role*), or a tree. For examples, the *climbing* multi-dimensional principle, applied to the Linear Precedence dimension (LP) and the Immediate Dominance dimension (ID), constrains the LP tree to be a flattening of the ID tree; and the *tree* principle applies to a single dimension and stipulates that this dimension must be a tree.

Configuration Task Our understanding of Duchier's statement according to which dependency parsing can be regarded as a *configuration problem* is as follows. The components are nodes and edges, where the nodes are the words from the input string and the edges are the roles which may connect words. The properties of these components are features such as, for the syntactic dimension, category, valency and agreement (number, gender, person) for the nodes, and role, modifier and modifiee for the edges. The constraints (in the grammar) specify how the nodes may be connected together by edges; they are the conditions of well-formedness. The desired configuration is a tree, which is specified by a set of specific constraints expressing “treeness” conditions.

Multiple Dimensions In XDG the initial model is extended to multiple *dimensions*, each dimension modelling a level of linguistic description—*e.g.* Linear Precedence, Predicate-Argument structure, Immediate Dominance, Scope structure, and Deep Syntax. Each dimension is formalised as a Constraint Satisfaction Problem(CSP) on its own, and the interaction among dimensions. In Debusmann, Duchier, and Kuhlmann (2005, p. 104), “each linguistic dimension is characterised using its own set of well-formedness conditions; interactions between different dimensions are specified at an interface level”.

As argued by Duchier, the multi-dimensional aspect of the formalism allows an integrated representation of language, since all dimensions can be represented within one single multi-graphs structure. Meanwhile, a drawback of such a representation comes from that the interaction between dimensions operates through universal principles. The inconvenient of universals appears quickly as soon as the grammar attempts to also tackle exceptions to these universals. Exceptions may either be dealt with by introducing inconsistencies in the constraint system, which will have to be dealt with by the solving process—this is typically the solution adopted in WCDG—, or it requires the principles to be increasingly more complex as the number of exceptions grows. To the best of our knowledge, this problem is not addressed in the literature about XDG.

Parsing Strategy An interesting aspect of XDG is that by being formalised as a configuration task, parsing is turned into a simple application case for existing solving strategies. Such a parser has been implemented in MOZART/OZ, which is a concurrent constraint programming environment.

Modelling Gradience Although XDG supports under-specification (Debusmann, Duchier, and Kuhlmann, 2005, §6.4)—the output being a set of partial graphs—two major criteria are still missing in XDG to make it, as it is, a model for gradience. One is the lack of weights associated with the constraints, and the other one is the lack of scoring function.

2.5.9 Estratat and Henocque

Different works from Estratat and Henocque (2004a,b) and Estratat (2006) describe PG parsing as a configuration task. Recall from §2.5.7 that a configuration task requires the architecture of the aimed artifact to be described. That is, the components and the way they connect together to form an instance of the artifact must be known. In the present case, the components are the constructions, and the connection among components is a constituency relationship. The components' properties (in the configuration sense of it), as well as the "connective" and the structural constraints, are the grammar's properties (in the PG sense of it). The goal is to build a *sentence*, which, therefore, is seen as the root element in the grammar. We will discuss later in this thesis the consequences that this goal has/yields on the way the grammar must be designed.

Practically, the configuration solver is implemented as an Object Oriented Constraint Programme (OOCP). Thus, the grammar is represented as an object model. In this model, constructions are *classes*, which inherit *feature* classes, the features being themselves organised into a *class hierarchy* to represent the feature structure associated with a construction. This model is well described in Estratat and Henocque (2005b,a).

A feature is mapped to a CSP variable in the model. As for the properties, as one can expect they are modelled as constraints, except the Constituency and Heads properties, which are relations between classes. Such a choice of representation raises at least a question, at most an issue. One aspect of PG, which makes it an interesting formalism, is that all types of (linguistic) properties can be dealt with in exactly the same way, unlike constituency-based formalism such as HPSG, where the Head relation is clearly privileged over the others. What is thus possible to do in PG is to relax a violated Head property and decide to still carry on with licensing the constituent. It turns out to be useful, for example, when a string contains an unknown gap. The situation may happen when parsing spoken language; a transcription of poor quality may contain gaps, which can not be clearly matched with a word from the lexicon. A similar situation occurs when parsing new or made up words (*i.e.* instances of *lexical openness* in Pullum and Scholz's taxonomy). Relaxing the Head constraint on a Noun, for example, may be useful in order to license a Noun Phrase. In Estratat and Henocque's model it is not clear to us what exactly the consequences of such a discrimination between the Constituency and Heads properties on one hand, and the other types of property on the other hand, are on the relaxation process and therefore on the licensing process. If the possibility to relax a constraint is quite intuitive, the one to relax a relation is not. Presumably, it is possible to include an isolated relation as a constraint in the CSP, and therefore the option to relax it is still open, but the reported description of the model prevents us from making a decision on this point.

Modelling Gradience Two key elements are missing in Estratat and Henocque's approach to qualify as a computational for gradience. One missing key element is the weighting of constraints. It should be clear by now that all constraints do no have same importance with respect to gradient grammaticality. Whether the framework could possibly be upgraded and adapted to include weighted constraints is not a question we have an answer to. The main question is to know whether OOCP supports the type of weighted CSPs, which would then be concerned.

Another missing element is a scoring function, necessary to model some ordering of candidate solutions, especially candidates partially violating the CSP. Estratat and Henocque do not report on the ability of their approach to include deviant components (*i.e.* violating part of the CSP) and elements of reasoning involving these components and a preference order over them. We will discuss these weaknesses further as part of the present thesis.

2.6 Conclusion

2.6.1 Summary

Accouting for Gradience in Language Study The problem of judging natural language acceptability involves performing a series of categorisation tasks, and making a series of decisions concerning the choice, for an item, of a host category among several candidates, and whether the item shows *enough* of the prototypical characteristics of a category. Even when a distinction applies between well-formed and deviant items (*e.g.* a horse *vs.* a three-legged horse) both situations are concerned with such decision-making issues. If linguistic theories have been paying a fair amount of attention to the former cases, they have also been failing to successfully address the latter—although changes in this regard were observed in recent years.

Even when acknowledging gradience in natural language, different standpoints are advocated. Ubiquitous gradience is one of them, where the clear-cut dichotomy between well-formedness and deviance is abandoned. Another one, advocated by Aarts, combines Aristotelian sharp boundaries for categories, assimilated to constructions, with a relaxed version of prototypicality where categorised items may not present all prototypical characteristics.

The Notion of Grammaticality The notion of grammaticality varies from one family of frameworks to another. Three main notions are identified: a generative notion, a model-theoretic notion, and an optimality-theoretic one.

The generative grammaticality, probably the most standard one in linguistic studies, is a strict binary notion whereby an utterance either belongs or not to the language.

The model-theoretic grammaticality comes closer to capture the intuitive and common-sense notion; it discriminates between grammatical expressions, which violate no constraints from the grammar, and ungrammatical quasi-expressions, which violate at least one constraint. The dual notion of ungrammaticality is graded, in the sense that a quasi-expression can be assigned a degree according to how many constraints from the grammar it violates.

The optimality-theoretic grammaticality applies to a *structure*, as opposed to an *utterance.* The grammatical structure for an utterance is the optimal one. The accepts variations and might be graded, in that no restriction is put on how many constraints are violated by a grammatical structure. Ungrammaticality, and *a fortiori* degrees of ungrammaticality, are not accounted for.

Knowledge Representation The question of the suitability of linguistic and computational frameworks to studying language and gradience in language was explored.

The comparison between Generative-Enumerative Syntax (GES) and Model-Theoretic Syntax (MTS) shows that an MTS framework is best capable of representing syntactic information for a large spectrum of linguistic phenomena, including cases of deviance. The general idea underpinning MTS is that a grammar is a set *statements* informing about linguistic properties of an utterance—properties which may either hold or fail. Similarly, Aarts relies on the same sort of statements, and suggests to categorise an item according to its *characteristic properties*. In OT, or LOT, the candidate structures are, again, ranked according to the constraints they violate.

Therefore, the notion of *constraint* appears to be playing a central role in approaches accounting for gradience. However, what kind of information is captured by constraints, and what kind of object it applies to differ substantially from one approach to another.

Computational Frameworks Following the preliminary conclusions we drew regarding requirements about knowledge representation for a computational account of gradience in natural language, different constraint-based frameworks were investigated.

The Constraint Dependency-based frameworks for language processing offer a descriptive approach to language study and a graded notion of constraint which comply to some of our requirements about knowledge representation. These frameworks adopt an optimality-theoretic notion of grammaticality.

The paradigm Property Grammars complies with all these requirements, and presents different features that enable the development of rating models for predicting the acceptability of an utterance. Most importantly, these features include an elegant and practical presentation of all the linguistic knowledge gathered about an utterance through the notion of *characterisation*. The characterisation of an utterance is a descriptive report on which of the constraints from the grammar are satisfied or violated by the utterance.

PG allows an implementation of construction Grammar, along with a model-theoretic notion of grammaticality and graded ungrammaticality. However, the existing implementations of parsing within PG show inconvenients, which are an impediment to the unbiased measurement of the relative influence of constraint types.

2.6.2 Pending Questions

These are the most important questions we had in mind all the way through the literature review, and which we keep in mind for the remainder of this dissertation. We aim to address them, and provide answers or elements of answers during the presentation of our works.

What Notion of Natural Language is Being Studied?

We share with Pullum and Scholz (and many other linguists) the idea that a natural language shows not only regularities and patterns of well-formedness, but also variations within well-formedness (*e.g.* among different local declinations), as well as different degrees of acceptability. It is such an "ordinary, common-sense" notion of natural language that we take as object of study.

What Notion of Grammaticality?

As we already emphasised it in several occasions (§§2.3.1, 2.3.2, 2.3.4, 2.6.1), being willing to find criteria on which to categorise a three-legged horse should not prevent us from the possibility to still acknowledge that a horse is a horse when we see one. Likewise in natural language: being able to account for a gradient of grammaticality should not prevent us from recognising and acknowledging well-formedness as well. Such an expectation is not met by optimality-theoretic models and alike (Keller, 2000; Menzel, 1995; Schröder et al., 2000; Schröder, 2002), where for a syntactic structure (of an input utterance) to be grammatical it is a matter of being better than all the other candidate structures—that is, it is a matter of being the *optimal candidate.* The problem is, that being the best candidate does not mean at all that the input is well-formed, and nothing is provided which would make it possible to make a clear-cut decision as to whether an utterance is well-formed or not.

Therefore, we prefer—and adopt—a model-theoretic notion of grammaticality with graded ungrammaticality, where classes of utterances are Constructions in the sense of CxG. Constructions should have sharp boundaries, with prototype members.

Is Gradience Modelled And How?

How are tackled the different issues related to modelling gradience and what are these related issues? This sums up the questions about how to bridge the gap between natural language and computational modelling of it. We share with Aarts the idea that natural language is made of categories with sharp boundaries and prototypes, but we do not rule out the possibility of overlapping categories and categories with fuzzy boundaries. We adopt a constructional approach to representing a language, with a model-theoretic representation of syntax, as it appears to be particularly well suited to process natural language as we conceive it.

How to Present and Represent the Linguistic Knowledge about an Utterance?

The main concern we have here is twofold:

- What kind of architecture is required for processing natural language in its ordinary notion?
- How to present the linguistic information gathered by the parsing process in a manner which is homogeneous regardless of the degree of grammaticality or ungrammaticality, and which provides all the information required for a degree of acceptability?

What Notion of Constraint?

The main questions we are asking about constraints are the following:

- What kind of information is captured and modelled by means of constraints, and how is it formally defined?
- What is the role played by constraints in the parsing process, if any?
- What sort of parsing strategies are adopted and what are their incentives?

What Role for the Satisfied Constraints?

Is acceptability affected by both violation and satisfaction of constraints, or does it only depend on violated constraints?

Many of the different accounts of gradience discussed here work under the assumption that only constraint violations are responsible for the degree of acceptability of a linguistic form, thus ignoring a possible influence of the "surrounding" satisfied constraints as well. This is, for instance, typically the case for approaches heavily inspired from, or influenced by Optimality Theory, for which constraint violation is quite central to language analysis (see §2.3.2 on Optimality Theory). Unfortunately, very little—if no evidence at all—is usually provided to sustain that assumption.

What we suggest to question here is whether satisfied constraints may also have an influence on acceptability judgements, along with the violated ones.

What scope for Constraint Weights?

The question we want to address here has to do with the spectrum of influence of constraint violations and satisfactions: does the violation/satisfaction of a specific constraint have same impact across constructions, or does the importance and the strength of the violation depend on the context it applies to?

To exemplify that question, let us consider a constraint of word order: does its violation entail same unacceptability whether it occurs in the context of a Noun Phrase, or in the context of Verb Phrase? In fact, several other questions are triggered simultaneously. One has to do with the granularity of the constraints: do we weight the type of constraint—*i.e.*, in this example, word order—or do we consider a more fine-grained level and assign different weights to a constraint of word order between a determiner and a noun, and to a constraint of word order between auxiliary and past participle? Even further: assuming that we assign a weight to the word order between determiner and noun, should it be the same weight whether the violation occurs in the context of a Subject-Verb Inversion construction, or in a Question construction, or in an affirmative statement?

What we are suggesting here is that violating a constraint (of word order between subject and verb, for that matter) may not have the same impact on the degree of acceptability of the input, depending on which construction is used to analyse the utterance.

What Kind of Scoring Function?

Quantifying acceptability can take different aspects, whether, for instance, involving intersective or subsective gradience, and also according to the type of syntactic structure that is dealt with. A frequent approach, though, is to resort to a scoring function, whose goal is to assess some aspect of gradience. Such scoring functions can take many different forms, and serve in fact different purposes, which we propose to investigate.

Therefore, the main questions we ask are:

- Where does a function come from? How was it designed, and how is it justified?
- Is the function linguistically motivated?
- What is the function used for: does it, for instance, compute an absolute degree of acceptability, or a relative one, with respect to some reference?

As we will see later, the development of a scoring function is an essential aspect of this thesis. We believe that, common to all the different accounts of gradience discussed in the literature so far, the role and the design of the scoring function have been greatly under-estimated.

Chapter 3

A Model-theoretic Logical Framework for Property Grammars

I can't get no satisfaction
I can't get no satisfaction
'Cause I try and I try and I try and I try
I can't get no, I can't get no

In *(I Can't Get No) Satisfaction*,
Mick Jagger and Keith Richards
for The Rolling Stones (1965)

3.1 Introduction

In Chapter 2 we came to the conclusion that neither GES nor OT provide suitable frameworks for representing and processing unrestricted natural language, while MTS is more appropriate. As a reminder, MTS aims to describe the syntactic properties of expressions, where *expression* is taken in the sense suggested by Pullum and Scholz, for whom it refers to an object linguists investigate. They cover sentences, clauses, phrases, words, idioms, lexemes, syllables, . . . from natural language. The grammar is a finite, unordered set of rules, where rules are independent statements about the well-formedness of expressions. The syntax of a well formed expression is described by a model satisfying the grammar.

An interesting aspect of MTS is that in spite of violating grammar rules, and since the grammar statements are independent from each other, a deviant expression may still be characterised by a *paraconsistent* model, which meets some of the grammar requirements and violates others. This property of MTS can be exploited so that the parsing (*i.e.* the building of a model) of an ill-formed input is no different from the parsing of a well-formed one, without resorting to any revision process.
The idea is in fact already implemented in Property Grammars (PG) (Blache, 2001, 2005), where each rule captures a syntactic property, and is represented as a constraint.

A formalisation of PG was proposed in VanRullen, Guénot, and Bellengier (2003); VanRullen (2005) (henceforth VRetAl), whose focus is to offer a modular reprensation of the different data structures involved in parsing with PG. Definitions are presented in BNF, along with the appropriate implementations formatted in DTD and XML. Interestingly the grammar is not only specified separately from the parsing process, but is also kept separate from the specification of its own core semantics. Specifically, the semantics of the different property types involved in the grammar is maintained separately from the grammar itself, which enables the specification of new property types without requiring modifications of the parsing process, or even modifications of the grammar itself. Of course, the introduction of new property types is usually motivated by specific needs with respect to the grammar, and therefore the grammar will be adapted accordingly with properties of the new type(s), but the specification of the property type semantics as such is independent from the parser and the grammar modules.
Following the body of works from Blache on PG the outcome of parsing an input utterance is a *characterisation* of this utterance. An input is characterised by a set of satisfied and violated properties. Definitions are provided in order to enable the specification of the conditions of satisfiability of a property (type), along with definitions for the characterisation of an input. These definitions underpin the formalisation provided for each of the different property types in use in VanRullen's thesis.

As far as the characterisation process is concerned, VRetAl also present a set of definitions underpinning the parsing process as implemented in VanRullen's thesis.

The *availability* of a property and subsequent notions, and the *satisfiability* of a property are the main notions introduced and defined. These notions are later referred to in the course of the description of the parsing process. However, VRetAl's works falls short of definitions for the constraint satisfaction and constraint relaxation mechanisms, and for the inference mechanisms associated with the *projection* of a category. Although definitions state that all the constraints in a PG grammar are *defeasible* (VanRullen, 2005, p. 82, Definition 24: Relâchement de contraintes (*Constraint Relaxation*)) and that in order to be projected a category must meet certain requirements (VanRullen, 2005, p. 85, Definition 30: Principe de Projectivité (*Projection Principle*)), neither is formally introduced *how* a constraint is relaxed, nor *how* a category is projected.

In this chapter we introduce a new logical framework based on First Order Predicate Logic (FOL) with Model-theoretic semantics, which lets us formalise the constraint satisfaction and constraint relaxation mechanisms, and the projection of a category. We have decided to define a new logical system rather than use and augment the one defined by VRetAl essentially for two reasons:

- in order to ease the introduction of quantifiers, and
- in order to avoid the manipulation of different types of variables, namely *énoncé* (*utterance*), *item*, and *token*.

In this new framework a constituent is seen as a *model* for a set of constraints. A constitent model is verified for a well-formed sentence by *strict satisfaction* of all the constraints in the grammar, while a *paraconsistent* model is verified for an ill-formed sentence by *loose satisfaction* of the grammar. *Projection rules* are also introduced, which allow the projection of the construction (labelled by a unique category) of a constituent through a mechanism inspired from typing judgements in programming languages. A formulation of the main property types in PG is also proposed.

§3.2 is dedicated to the formal definition of a new logical system for PG, and §3.3 discusses different salient properties of the framework. §3.4 closes the chapter with a conclusion.

3.2 A Logical System for Property Grammars

What we aim to define is a logical system (called Ξ) in which an MTS grammar for a given natural language can be specified, and which allows reasonning about any utterance from this language. Unlike in GES where an utterance from Natural Language is represented on the syntactic level by a string of terminals from the logical language, in MTS an utterance is represented on the semantic level. The logical language in use relies on a standard First-Order Logic (FOL) baseline, and allows specifying requirements of well-formedness about objects in the domain as formulae. After defining syntactic and semantic rules for Ξ we focus on the formalisation of PG in Ξ. We propose a formulation for a PG grammar and for the main constraint types. We then define a non-classical semantic consequence in order to substantiate the projection mechanism in PG.

3.2.1 Syntax

Language The language $\mathcal{L}$ we use in Ξ contains:

- constant symbols
- variables
- logical connectives, with equality $(\neg, \wedge, \vee, =, \ldots)$;
- predicate and function symbols;
- True and False symbols (respectively $\top$ and $\bot$).

Definition 3.1 (Term) *There exists three sorts of terms:*

1. *individual constants,*
2. *individual variables,*
3. *composite terms, such as $f(x), f(g(x)), \ldots$, where f and g are function symbols.*

Definition 3.2 (Atomic Formula) *An* atomic formula φ *is such that either*

1. $\varphi := t_1 = t_2$*, where* t_1 *and* t_2 *are terms, or*

2. $\varphi := R(t_1, \ldots, t_n)$*, where* R *is an* n*-ary relation symbol.*

Formula A *well-formed formula* (wff) is a well-formed sentence of $\mathcal{L}$. We sometimes denote by $\phi(x_1, \ldots, x_n)$ the formula ϕ over the variables $(x_1, \ldots, x_n)$.

3.2.2 Semantics

Ξ relies on standard FOL semantics; we use defintions from Gochet and Gribomont (1990) and Shapiro (Fall 2007).

Domain of Discourse

We already quoted Pullum and Scholz (p. 16) for whom expressions from natural language are taken

> to *have* syntactic structure, not merely to be analyzable in terms of structures imposed on them or posited for them by linguists. (Pullum, 2007)

Therefore it is precisely this syntactic structure that the semantic objects have in the domain $\mathcal{D}$, and which we aim to reason about in Ξ. It is this structure that we describe now.

Individuals All individuals in $\mathcal{D}$ are *constituents*. From a linguistic point of view, a constituent is a representation of a generalised expression and may be a sentence, a clause, a phrase, a word, an idiom, a lexeme, a syllable, ... A constituent is specified by:

- A collection of *features*, represented as an Attribute-Value Matrix (AVM). This feature structure contains at least:
  - An identifying label
  - The feature CAT, whose value is a reference to the class of constituents this constituent belongs to. A class of constituents is called a *construction*.

- a possibly empty set of sub-constituents, represented as an ordered list of references to constituents in $\mathcal{D}$.

In PG, it is suggested that the structure of a constituent be even more informative by also containing its *characterisation* in terms of syntactic relationships among its sub-constituents. We discuss that aspect later, but we do not make it a requirement in the definition of a constituent.

Lexical Item An atomic constituent, which has no sub-constituent, is a lexical item (*i.e.*, a word). For the sake of this thesis we assume that it is an inflected word, since we are not directly concerned by the morphological level of the linguistic study. But it is possible, according to the level of linguistic investigation desired, to further decompose a word in more fine-grained sub-constituents such as syllable, stem, case-marker, ... A word in the lexicon is specified as any other constituent with a feature structure, but unlike non-atomic constituents it is not specified by any constraints. Examples of features are GENDER, and NUMBER.

Lexicon What is usually defined as the *lexicon* is, in Ξ, no different from the grammar, which we are going to define shortly. From a linguistic point of view, such a conception of a lexicon is in line with the one in Construction Grammar, as suggested for instance by Goldberg (1995). A word from natural language is nothing but a specific (lexical) construction, and an uttered word is analysed as an instance of that construction.

Definition 3.3 (Interpretation) *An* Interpretation *for the language $\mathcal{L}$ is the tuple $\mathcal{S} = \langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is the domain of discourse, and $\mathcal{I}$ an* interpretation function *which maps each non-logical symbol to a value in $\mathcal{D}$ as following Gochet and Gribomont (1990, p. 295):*

1. *each individual constant c is mapped to a member of $\mathcal{D}$, noted $\|c\|_{\mathcal{I}} = c^{\mathcal{I}}$;*

2. *each n-ary function symbol f is mapped to a function, which takes its arguments in $\mathcal{D}^n$ and its values in $\mathcal{D}$;*

3. *each n-ary predicate symbol is mapped to an n-ary relation, i.e. a class of tuples members of $\mathcal{D}^n$.*

Definition 3.4 (Assignment) *An* assignment $\mathcal{A}$ *on an interpretation structure* $\mathcal{S}$ *is a mapping from a set of variables to values in the domain* $\mathcal{D}$ *of* $\mathcal{S}$*.*

We note $\mathcal{A} = \langle \|x_1\|_{\mathcal{A}} = V_1, \|x_2\|_{\mathcal{A}} = V_2, \ldots, \|x_n\|_{\mathcal{A}} = V_n \rangle$, or $\mathcal{A} = \langle V_1, V_2, \ldots, V_n \rangle$ for short, to refer to the tuple of values from the range $\mathcal{D}^n$ which are assigned to variables.

Constants are interpreted with the *interpretation* function $\mathcal{I}$, while variables are interpreted with the *assignment* function $\mathcal{A}$, and composite terms require both $\mathcal{I}$ and $\mathcal{A}$. Intuitively, an assignment is made up of an individual assigned to a root constituent, and a sequence of contiguous[1] individuals assigned to the sub-constituents. For example, the assignment $\langle \|x_1\|_{\mathcal{A}} = \mathrm{NP}_1, \|x_2\|_{\mathcal{A}} = \mathrm{D}_2, \|x_3\|_{\mathcal{A}} = \mathrm{N}_3 \rangle$ assigns the embedding constituent NP_1 to x_1, while x_2 and x_3 are assigned respectively the Part-of-Speech constituents D_2 (determiner) and N_3 (noun). D_2 and N_3 are sub-constituents of NP_1; they are given in $\mathcal{A}$ in the same order as in the sentence they appear in.

Semantic Rules for Terms and Formulae

We refer here to standard rules, as stated in Gochet and Gribomont (1990, pp. 298, 300–301), for interpreting terms and formulae:

SR1 for a constant of individual c, $\|c\|_{\mathcal{I},\mathcal{A}} = \|c\|_{\mathcal{I}} = c^{\mathcal{I}}$;

SR2 for a variable v, $\|v\|_{\mathcal{I},\mathcal{A}} = \|v\|_{\mathcal{A}}$;

SR3 for a function symbol f, $\|f(t_1, \ldots, t_n)\|_{\mathcal{I},\mathcal{A}} = \|f\|_{\mathcal{I}} (\|t_1\|_{\mathcal{I},\mathcal{A}}, \ldots, \|t_n\|_{\mathcal{I},\mathcal{A}})$, where t_i are terms;

SR4 $\|P(t_1, \ldots, t_n)\|_{\mathcal{I},\mathcal{A}} = \top$ iff $\langle t_1^{\mathcal{I},\mathcal{A}}, \ldots, t_n^{\mathcal{I},\mathcal{A}} \rangle \in P^{\mathcal{I}}$, where t_i are terms and $t^{\mathcal{I},\mathcal{A}}$ their denotations;

[1] In Chapter 4 we introduce various restrictions for implementing this framework. Assumption 5 requires the contiguity of the sub-constituents of a constituent, which is captured in $\mathcal{A}$ by the contiguity of the individuals.

SR5 $\|\neg X\|_{\mathcal{I},\mathcal{A}} = \top$ iff $\|X\|_{\mathcal{I},\mathcal{A}} = \bot$;

SR6 $\|X \wedge Y\|_{\mathcal{I},\mathcal{A}} = \top$ iff $\|X\|_{\mathcal{I},\mathcal{A}} = \top$ and $\|X\|_{\mathcal{I},\mathcal{A}} = \top$;

SR7 $\|\forall x\phi\|_{\mathcal{I},\mathcal{A}} = \top$ iff $\phi^{\mathcal{I},\mathcal{B}} = \top$ for all assignment $\mathcal{B}$ identical to $\mathcal{A}$, except *possibly* for the individual assigned to x by $\mathcal{B}$ ($\mathcal{B}$ is called an *x–variant* of $\mathcal{A}$);

SR8 $\|\exists x\phi\|_{\mathcal{I},\mathcal{A}} = \top$ iff $\phi^{\mathcal{I},\mathcal{B}} = \top$ for at least one x–variant $\mathcal{B}$ of $\mathcal{A}$.

3.2.3 Formulation of the PG Constraint Types

From a linguistic point of view, constraints in PG express linguistic statements such as:

- A Determiner precedes a Noun (*e.g.* (P3.11))
- A Noun is unique (*e.g.* (P3.8))
- The presence of either a Noun or a Pronoun is compulsory (*e.g.* (P3.6))

A *constraint* is a well-formed formula in Ξ. In PG, several template formulae are predefined, and these are called *constraint types*. A constraint type is *generic*, in that it applies not only to variables but also to the constructions of these variables. For instance, a constraint from the generic type *Linearity* holds between two constituents, with each constituent being of a specific construction (*e.g.* linearity between a Determiner and a Noun). Therefore the constructions are seen as *parameters* of the constraint type. A constraint type is thus defined as a multi-parametered formula, where a parameter is always a construction.

Each constraint type is associated with a specific (multi-parametered) relation in Ξ. Below we use the following notation; for each constraint type its parameter category labels are put in square brackets right after the predicate label, while variables are put in parentheses (*e.g.* linearity$[\textsc{cat}_1, \textsc{cat}_2](r)$). The variables in parentheses are given to allow references out of the scope of the formula. Equivalent abbreviated notations are also provided for each type, which omit the variables (*e.g.* $\textsc{cat}_1 \prec \textsc{cat}_2$). These abbreviated notations are those more commonly used in the literature about PG.

In this study, we use 6 pre-defined constraint types: *linearity*, *obligation*, *uniqueness*, *requirement*, *exclusion*, and *dependency*, which we define below.

Pre-requisites

Special Predicates Various special predicates are used in the formulation of the constraint types and across the present study. Their semantics is meant to be intuitive:

- The relation $\text{constituents}(x, y_1, \ldots, y_n)$ denotes that the constituents $\|y_1\|_{\mathcal{I},\mathcal{A}}, \ldots, \|y_n\|_{\mathcal{I},\mathcal{A}}$ are immediate (sub–)constituents of the dominant constituent $\|x\|_{\mathcal{I},\mathcal{A}}$.
  The relation is not exclusive; that is $y_1, \ldots, y_n$ need not be the only constituents of x.

- The function $\text{right}(x)$ (respectively $\text{left}(x)$) returns the position index in the input string of the right end (left end) of x.

- The function . (*dot*), as in the term $x.f$, returns the value of the feature f in x.

- When necessary, the feature structure of a constituent is given as an AVM, sometimes next to the term it is associated with (*e.g.* $\text{N}\begin{bmatrix}\text{GEND } \boxed{1} \\ \text{NUM } \boxed{2}\end{bmatrix}$)

Typed Feature Structure It may be useful, in some cases, to provide the possibility to constrain not only the constructions of constituents, but also their feature structures. For that purpose, it is necessary to enable the specification of a description of a feature structure. The logic of typed feature structures is now well-established (Carpenter, 1992; Kasper and Rounds, 1986); we thus only give here the minimum definitions required for the description of flat AVMs and standard unification. Note that we do not include any default unification operations (Bouma, 1992; Lascarides and Copestake, 1999), following in that one of the original motivations for PG whereby all constraints in the grammar play equal roles and are, therefore, all defeasible. It is not clear, at this stage, how the prioritisation of constraints through the use of defaults in feature structures would interact with loose satisfaction, which we introduce later on (see §3.2.5) in order to capture constraint relaxation in PG. We acknowledge that there is an avenue for further investigation here, in light, for instance, of recent works from Malouf (2007) where a heuristic-based optimisation is presented for default unification that can be used to prune the search space.

Using the Backus-Naur Form (BNF) notation, a *Description* D is defined by:

$$D := x.f = v \mid x.f_1 = y.f_2 \mid D\&D$$

with the following semantics:

$$\mathcal{I}, \mathcal{A} \models D_1 \& D_2 \qquad \text{iff } \mathcal{I}, \mathcal{A} \models D_1 \text{ and } \mathcal{I}, \mathcal{A} \models D_2 \tag{3.1}$$

$$\mathcal{I}, \mathcal{A} \models x.f = v \qquad \text{iff } \mathcal{A}(x).f = I(v) \tag{3.2}$$

$$\mathcal{I}, \mathcal{A} \models x.f_1 = y.f_2 \qquad \text{iff } \mathcal{A}(x).f_1 = \mathcal{A}(y).f_2 \tag{3.3}$$

This way, the Description involved, for instance, in (P3.29) is equivalent to the following expression:

$$x.\text{PERS} = y.\text{PERS} \tag{3.4}$$

$$\& \; x.\text{NUM} = y.\text{NUM} \tag{3.5}$$

$$\& \; y.\text{TYPE} = pers \tag{3.6}$$

$$\& \; y.\text{CASE} = nom \tag{3.7}$$

We use the predicative notation $\text{Desc}(x', y') : D$ to denote a two-argument Description expressed in the language D previously defined; we use $D[x/x', y/y']$ to denote the result of substituting x for x' and y for y' in D, and we use $D|_x$ to denote the restriction of D to only those literals which mention no variables other than x.

Quantifiers We extend $\mathcal{L}$ with the new quantifier Λ with the following semantics:

$$\mathcal{I}, \mathcal{A} \models \Lambda x : \phi \quad \text{iff} \quad \forall x' \in \text{dom}(\mathcal{A}) \;\; \mathcal{I}, \mathcal{A} \models \phi[x'/x] \tag{3.8}$$

where $\mathrm{dom}(\mathcal{A})$ denotes the subset of the domain $\mathcal{D}$ made up of the individuals in $\mathcal{A}$. The idea, here, is to quantify only over the variables of the assignment $\mathcal{A}$. The purpose of Λ, in the following definitions, is to maintain closed formulae. The semantics of the existential quantifier $\exists$ remains unchanged.

Definitions

In the following definitions of each constraint, two formulations are provided: a general one, with no Descriptions among the parameters, and a more fine-grained one, which requires a parameter Description.

Definition 3.5 (Linearity) *A constraint of* linearity*, denoted by* $\mathrm{linearity}[\mathrm{C}_1, \mathrm{C}_2](r)$ *or* $\mathrm{linearity}[\mathrm{C}_1, \mathrm{C}_2, \mathrm{Desc}(x', y') : D](r)$ *is defined in* Ξ *by the following formulae:*

$$\mathrm{linearity}[\mathrm{C}_1, \mathrm{C}_2](r) \textit{ iff}$$
$$\Lambda x \Lambda y\, \mathrm{constituents}(r, x, y) \wedge x.\textsc{cat} = \mathrm{C}_1 \wedge y.\textsc{cat} = \mathrm{C}_2 \rightarrow \mathrm{right}(x) < \mathrm{left}(y) \quad (3.9)$$

$$\mathrm{linearity}[\mathrm{C}_1, \mathrm{C}_2, \mathrm{Desc}(x', y') : D](r) \textit{ iff}$$
$$\Lambda x \Lambda y\, \mathrm{constituents}(r, x, y) \wedge x.\textsc{cat} = \mathrm{C}_1 \wedge y.\textsc{cat} = \mathrm{C}_2 \wedge D[x/x', y/y'] \rightarrow \mathrm{right}(x) < \mathrm{left}(y) \quad (3.10)$$

The alternative notations are $\mathrm{C}_1 \prec \mathrm{C}_2$*, or* $\mathrm{lin}(\mathrm{C}_1, \mathrm{C}_2)$*.*

A constraint of linearity specifies an ordering relation between constituents, in terms of the word order in the input string.

Definition 3.6 (Requirement) *A constraint of* requirement*, denoted by* $\mathrm{requirement}[\mathrm{C}_1, \mathrm{C}_2](r)$ *or* $\mathrm{requirement}[\mathrm{C}_1, \mathrm{C}_2, \mathrm{Desc}(x', y') : D](r)$ *is defined in* Ξ *by* *the following formulae:*

$$\begin{aligned}&\text{requirement}[\mathrm{C}_1,\mathrm{C}_2](r)\ \textit{iff}\\ &\qquad\qquad \Lambda x\,\text{constituents}(r,x)\wedge x.\textsc{cat}=\mathrm{C}_1\\ &\qquad\qquad\qquad \rightarrow \exists y\,\text{constituents}(r,y)\wedge y.\textsc{cat}=\mathrm{C}_2\end{aligned}\tag{3.11}$$

$$\begin{aligned}&\text{requirement}[\mathrm{C}_1,\mathrm{C}_2,\text{Desc}(x',y'):D](r)\ \textit{iff}\\ &\qquad \Lambda x\,\text{constituents}(r,x)\wedge x.\textsc{cat}=\mathrm{C}_1\wedge D|_x[x/x']\\ &\qquad\qquad \rightarrow \exists y\,\text{constituents}(r,y)\wedge y.\textsc{cat}=\mathrm{C}_2\wedge D[x/x',y/y']\end{aligned}\tag{3.12}$$

The alternative notations are $\mathrm{C}_1 \Rightarrow \mathrm{C}_2$*, or* $\text{req}(\mathrm{C}_1,\mathrm{C}_2)$*.*

A constraint of requirement specifies a property of co-occurrence between constituents. The co–occurrence is not symmetric[2], in that $\textsc{cat}_1 \Rightarrow \textsc{cat}_2$ is *not* equivalent to $\textsc{cat}_2 \Rightarrow \textsc{cat}_1$.

Definition 3.7 (Exclusion) *A constraint of* exclusion*, denoted by* $\text{exclusion}[\mathrm{C}_1,\mathrm{C}_2](r)$ *or* $\text{exclusion}[\mathrm{C}_1,\mathrm{C}_2,\text{Desc}(x',y'):D](r)$ *is defined in* Ξ *by the following formulae:*

$$\begin{aligned}&\text{exclusion}[\mathrm{C}_1,\mathrm{C}_2](r)\ \textit{iff}\\ &\quad (\Lambda x\,\text{constituents}(r,x)\wedge x.\textsc{cat}=\mathrm{C}_1\rightarrow\neg\exists y\,\text{constituents}(r,y)\wedge y.\textsc{cat}=\mathrm{C}_2)\\ &\quad\ \wedge(\Lambda u\,\text{constituents}(r,u)\wedge u.\textsc{cat}=\mathrm{C}_2\rightarrow\neg\exists v\,\text{constituents}(r,v)\wedge v.\textsc{cat}=\mathrm{C}_1)\end{aligned}\tag{3.13}$$

[2] It was pointed out to us that *conditionality* might be a more appropriate description than *co-occurrence*, as co-occurrence is usually perceived as being symmetric. We keep the reference to co-occurrence to match the literature about PG in that regard.

$$\begin{aligned}
&\mathrm{exclusion}[\mathrm{C}_1, \mathrm{C}_2, \mathrm{Desc}(x', y') : D](r) \textit{ iff} \\
&\qquad (\Lambda x\, \mathrm{constituents}(r, x) \wedge x.\textsc{cat} = \mathrm{C}_1 \wedge D[x/x'] \\
&\qquad \rightarrow \neg\exists y\, \mathrm{constituents}(r, y) \wedge y.\textsc{cat} = \mathrm{C}_2) \wedge D[y/y'] \\
&\qquad \wedge (\Lambda u\, \mathrm{constituents}(r, u) \wedge u.\textsc{cat} = \mathrm{C}_2 \wedge D[u/u'] \\
&\qquad\qquad \rightarrow \neg\exists v\, \mathrm{constituents}(r, v) \wedge v.\textsc{cat} = \mathrm{C}_1 \wedge D[v/v'])
\end{aligned} \tag{3.14}$$

The alternative notations are $\mathrm{C}_1 \not\Leftrightarrow \mathrm{C}_2$*, or* $\mathrm{excl}(\mathrm{C}_1, \mathrm{C}_2)$*.*

A constraint of exclusion prevents a co–occurrence between two constructions. The relation is symmetric.

Definition 3.8 (Obligation) *A constraint of* obligation*, denoted by* $\mathrm{obligation}[\textsc{cat}_1](r)$ *or* $\mathrm{obligation}[\mathrm{C}_1, \mathrm{Desc}(x') : D](r)$*, is defined in* Ξ *by the following formulae:*

$$\begin{aligned}
&\mathrm{obligation}[\mathrm{C}_1](r) \textit{ iff} \\
&\qquad \Lambda x\, \mathrm{constituents}(r, x) \rightarrow \exists y\, \mathrm{constituents}(r, y) \wedge y.\textsc{cat} = \mathrm{C}_1
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
&\mathrm{obligation}[\mathrm{C}_1, \mathrm{Desc}(x') : D](r) \textit{ iff} \\
&\qquad \Lambda x\, \mathrm{constituents}(r, x) \rightarrow \exists y\, \mathrm{constituents}(r, y) \wedge y.\textsc{cat} = \mathrm{C}_1 \wedge D[x/x']
\end{aligned} \tag{3.16}$$

The alternative notations are $\triangle\mathrm{C}_1$ *or* $\mathrm{obl}(\mathrm{C}_1)$*.*

A constraint of obligation specifies a required construction. Obligation is typically used to specify heads.

Definition 3.9 (Uniqueness) *A constraint of* uniqueness*, denoted by* $\mathrm{uniqueness}[\mathrm{C}_1]()$ *or* $\mathrm{uniqueness}[\mathrm{C}_1, \mathrm{Desc}(x') : D]()$ *is defined in* Ξ *by the following formulae:*

$$\begin{aligned}
&\mathrm{uniqueness}[\mathrm{C}_1](r) \textit{ iff} \\
&\qquad \Lambda x \Lambda y\, \mathrm{constituents}(r, x, y) \wedge x.\textsc{cat} = \mathrm{C}_1 \wedge y.\textsc{cat} = \mathrm{C}_1 \\
&\qquad\qquad \rightarrow x = y
\end{aligned} \tag{3.17}$$

$$\begin{aligned}&\mathrm{uniqueness}[\mathrm{C}_1, \mathrm{Desc}(x') : D](r) \textit{ iff}\\ &\qquad \Lambda x \Lambda y\, \mathrm{constituents}(r, x, y) \wedge x.\textsc{cat} = \mathrm{C}_1 \wedge y.\textsc{cat} = \mathrm{C}_1 \wedge D[x/x']\\ &\qquad\qquad \rightarrow x = y \wedge D[y/x'] \end{aligned} \tag{3.18}$$

The alternative notations are $\mathrm{C}_1!$*, or* $\mathrm{uniq}(\mathrm{C}_1)$*.*

A constraint of uniqueness specifies the presence of a unique constituent from a given construction.

The Dependency Constraint Type The *dependency* constraint type is slightly different from the other types, in that it is used for stating a semantic relationship between objects rather than for checking syntactic conditions. In Blache (2005), it is said to

> (...) stipulate[s] a semantic dependency between different objects of the construction. (...) On top of this role, dependency is also used in order to implement semantic restrictions on the argument structure. This is done directly by specifying the corresponding semantic feature in the restricted object.

In practice, dependency is used, for instance, to enforce agreements (*e.g.* person, or gender).

Definition 3.10 (Dependency) *A constraint of* dependency*, denoted by* $\mathrm{dependency}[\mathrm{C}_1, \mathrm{C}_2, \mathrm{Desc}(x', y') : D](r)$ *is defined in* Ξ *by the following formula:*

$$\begin{aligned}&\mathrm{dependency}[\mathrm{C}_1, \mathrm{C}_2, \mathrm{Desc}(x', y') : D](r) \textit{ iff}\\ &\qquad \Lambda x \Lambda y\, \mathrm{constituents}(r, x, y) \wedge x.\textsc{cat} = \mathrm{C}_1 \wedge y.\textsc{cat} = \mathrm{C}_2\\ &\qquad\qquad \rightarrow D[x/x', y/y'] \end{aligned} \tag{3.19}$$

The alternative notations are $\mathrm{C}_1 \rightsquigarrow \mathrm{C}_2$*, or* $\mathrm{dep}(\mathrm{C}_1, \mathrm{C}_2)$*.*

Note that the symbol $\rightsquigarrow$ commonly in use in the literature about PG might be slightly misleading since the relation is in fact symmetric.

3.2.4 Grammar

The purpose of an MTS grammar is to specify conditions of well-formedness on the objects from $\mathcal{D}$. In PG, a condition of well-formedness is local to the context of a specific class of utterances, called a *construction*. Therefore a PG grammar is a collection of constructions, where a construction is specified by a conjunction of constraints. Tables 3.2 and 3.3 gives an example of a PG grammar.

An important aspect of PG is that a constraint is not *universal*, but is a condition of well-formedness which only applies to the instances of the construction it specifies. In that sense, it is a *membership requirement*, which allows the classification of an utterance when met. The *projection* process in PG specifies that from any constraint in the grammar a constituent can be inferred (*i.e. projected*) of the category this constraint specifies. Therefore, in Ξ each PG constraint is paired in the grammar with a deduction rule—called a *projection rule*. A projection rule is of the form $\Lambda r\ r.\textsc{cat} = c \rightarrow \phi(r)$. It specifies a constraint, which must be satisfied by all the constituents of the model. It is trivially satisfied for those constituents whose construction in not c; as for those whose construction is c, then $\phi(r)$ must be satisfied as well. In other words, the projection rules enable judgements about the construction of a constituent.

Definition 3.11 (Grammar) *A PG grammar in Ξ is defined as a set of pairs $\langle\phi(r), \psi\rangle$, where:*

- *$\phi(r)$ is an atomic formula in $\mathcal{L}$,*
- *ψ is a projection rule such that $\Lambda r\ r.\textsc{cat} = cat \rightarrow \phi(r)$, and*
- *cat is a construction label.*

For instance, the fact that the construction labelled c is specified by the constraint $\phi(r) := \text{linearity}[\textsc{cat}_1, \textsc{cat}_2](r)$ is represented by the pair $\langle\phi(r), \psi\rangle$ with $\psi := \Lambda r\ r.\textsc{cat} = c \rightarrow \phi(r)$.

In §4.4.5 we describe different lookup tables that we use in order to consult the grammar. One of these tables, the Projection Index, is a representation of all the projection rules in the grammar, where a rule's right-hand side is indexed by the rule's left-hand side.

Toy Grammar We have developed a PG Construction Grammar for French, based on the one developed by VanRullen et al. (2005). Their grammar was implemented to participate in the EASY (Évaluation d 'Analyseurs SYntaxiques) evaluation campaign for parsers of French (Gendner et al., 2003; Paroubek, Robba, and Ayache, 2007). Its implementation meets the requirements provided as part of the evaluation protocol, and only uses flat constituents—*i.e.* without any nested constituents. We have adapted it and extended it in order to handle constituent structures with nested constituents. We have done so essentially (but not exclusively) by making, next to VanRullen's EASY grammar, several borrowings to other existing PG grammars, sometimes allowing ourselves a rather free interpretation of them in our own implementation: the one developed by Guénot (2006), and the one developed by Blache, Hemforth, and Rauzy and underpinning the experiments presented in their 2006 paper. We have also complemented these with various other rules inspired from Riegel, Pellat, and Rioul (1998), in order to make it a consistent core set of grammar rules for French[3]. The full grammar comprises of 16 constructions including 8 Part-of-Speech

[3] We would like to stress that the development of a grammar for French has not been central to this work. As a consequence, different imperfections may have slipped in. Indeed, and despite these references of us discrepencies remain, and it goes without saying that these should only be blamed on us. A few of those were pointed out to us by Eric Villemonte de la Clergerie (examiner, *examinateur*), which we would like to address briefly.

• In S, the Dependency constraint (P3.5) does not involve any AVMs: this is not exactly a discrepency, but it is asking for an explanation. This constraint does not hold on any features; the dependency relationship between the two constructions is simply asserted (see the paragraph on the Dependency Constraint Type, p. 80, for more details), thus the constraint holds when the two constructions NP and VP are present in a constituent from S.

• In NP, the Uniqueness constraint (P3.9) is inaccurate: in practice, the PP-attachements are dealt with separately from the NP construction. This kind of discrepencies are due to late evolutions that occurred during the experimental phase.

• In NP, a Dependency constraint $\mathrm{AP} \leadsto \mathrm{N}$ is missing.

• In VP, the Exclusion constraint (P3.28) means that in case of two clitics (one of each case) one has to be dealt with out of the VP.

• In AP, the Uniqueness constraint (P3.33) is arguable, as an AP may contain more than one adverb; in that case, we make them as many APs.

• In AP, the Uniqueness of a past participle $\mathrm{V}_{[past\,part.]}!$ ((P3.32)) could be generalised and replaced by an $\mathrm{S}_{[past\,part.]}!$. The problem here is that feature inheritance being not implemented, the categories, and 100 constraints. A brief overview of it, as well as of the lexicon, is reported in §4.9.3. The full grammar is provided in Appendix A. A sample (Γ) of it, presented in Tables 3.2 and 3.3, gives a flavour of the full-size implementation. The legend is presented in Table 3.1 below[4]. This sample is used as the underpinning grammar in various examples across this dissertation.

Symbol	Convention
*	Identifies a deviant constituent, which violates at least one constraint
star	Used as a wildcard, when no spanning parse could be found
D	Determiner
N	Noun
V	Verb
A	Adjective
Adv	Adverb
P	Preposition
Pro	Pronoun
C	Conjunction
NP	Noun Phrase
VP	Verb Phrase
PP	Prepositional Phrase
AP	Adjective Phrase
S	Utterance

Table 3.1: Legend and graphic conventions used in this dissertation

construction S does not have any features attached to it.

- In PP, the Requirement constraint (P3.42) is inaccurate; given (P3.41) it should rather make either an NP *or* a VP a requirement.
- In PP again, about (P3.43): since VPs are allowed by (P3.41), it would be more accurate to also have a P $\rightsquigarrow$ NP.

[4] Note that these labels are actually the English version of the real tagset in use, where the labels are in French. We thought that given how often these labels are referred to in this dissertation using English terms would ease the reading.

S (*Utterance*)		
Features	Property Type : Properties	
[AVM]	*obligation* : $\triangle$VP	(P3.1)
	uniqueness : NP!	(P3.2)
	: VP!	(P3.3)
	linearity : NP $\prec$ VP	(P3.4)
	dependency : NP $\rightsquigarrow$ VP	(P3.5)

NP (*Noun Phrase*)		
Features	Property Type : Properties	
[AVM]	*obligation* : *Obl*(N $\vee$ Pro)	(P3.6)
	uniqueness : D!	(P3.7)
	: N!	(P3.8)
	: PP!	(P3.9)
	: Pro!	(P3.10)
	linearity : D $\prec$ N	(P3.11)
	: D $\prec$ Pro	(P3.12)
	: D $\prec$ AP	(P3.13)
	: N $\prec$ PP	(P3.14)
	requirement : N $\Rightarrow$ D	(P3.15)
	: AP $\Rightarrow$ N	(P3.16)
	exclusion : N $\not\Leftrightarrow$ Pro	(P3.17)
	dependency : $\mathrm{N}\begin{bmatrix}\textsc{gend} & \boxed{1}\\ \textsc{num} & \boxed{2}\end{bmatrix} \rightsquigarrow \mathrm{D}\begin{bmatrix}\textsc{gend} & \boxed{1}\\ \textsc{num} & \boxed{2}\end{bmatrix}$	(P3.18)

VP (*Verb Phrase*)		
Features	Property Type : Properties	
[AVM]	*obligation* : $\triangle$V	(P3.19)
	uniqueness : $\mathrm{V}_{[main\,past\,part.]}$!	(P3.20)
	: NP!	(P3.21)
	: PP!	(P3.22)
	linearity : V $\prec$ NP	(P3.23)
	: V $\prec$ Adv	(P3.24)
	: V $\prec$ PP	(P3.25)
	requirement : $\mathrm{V}_{[past\,part.]} \Rightarrow \mathrm{V}_{[aux.]}$	(P3.26)
	exclusion : $\textsc{Pro}_{[acc]} \not\Leftrightarrow \mathrm{NP}$	(P3.27)
	: $\textsc{Pro}_{[dat]} \not\Leftrightarrow \textsc{Pro}_{[acc]}$	(P3.28)
	dependency : $\mathrm{V}\begin{bmatrix}\textsc{pers} & \boxed{1}\\ \textsc{num} & \boxed{2}\end{bmatrix} \rightsquigarrow \textsc{Pro}\begin{bmatrix}\textsc{type} & pers\\ \textsc{case} & nom\\ \textsc{pers} & \boxed{1}\\ \textsc{num} & \boxed{2}\end{bmatrix}$	(P3.29)

Table 3.2: Γ: An Example PG Grammar for French (1)

AP (*Adjective Phrase*)

Features	Property Type : Properties	
[AVM]	*obligation* : $Obl(\mathrm{A} \vee \mathrm{V}_{[past\,part.]})$	(P3.30)
	uniqueness : A!	(P3.31)
	: $\mathrm{V}_{[past\,part.]}$!	(P3.32)
	: Adv!	(P3.33)
	linearity : A $\prec$ PP	(P3.34)
	: Adv $\prec$ A	(P3.35)
	exclusion : A $\not\Leftrightarrow$ $\mathrm{V}_{[past\,part.]}$	(P3.36)

PP (*Prepositional Phrase*)

Features	Property Type : Properties	
[AVM]	*obligation* : $\triangle$P	(P3.37)
	uniqueness : P!	(P3.38)
	: NP!	(P3.39)
	linearity : P $\prec$ NP	(P3.40)
	: P $\prec$ VP	(P3.41)
	requirement : P $\Rightarrow$ NP	(P3.42)
	dependency : P $\rightsquigarrow$ NP	(P3.43)

Table 3.3: Γ: An Example PG Grammar for French (2)

3.2.5 Satisfaction

Definition 3.12 (Satisfaction (strict)) *Let ϕ be a formula of $\mathcal{L}$; the tuple $\langle \mathcal{I}, \mathcal{A} \rangle$ is said to* strictly satisfy *ϕ iff $\|\phi\|_{\mathcal{I},\mathcal{A}} = \top$. We note*

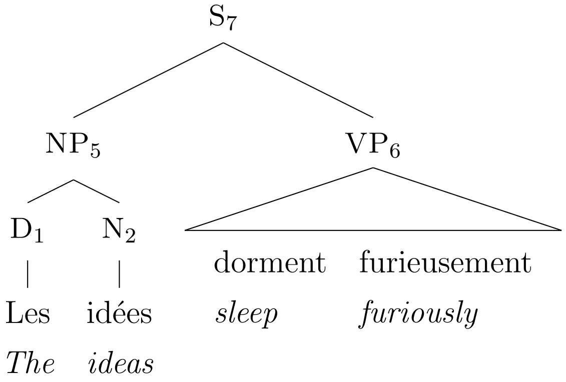
$$\mathcal{I}, \mathcal{A} \models \phi$$

The tuple $\langle \mathcal{I}, \mathcal{A} \rangle$ such that $\mathcal{I}, A \models \phi$ is called a strict model *for ϕ.*

Example 3.1 (Model Checking) *Let us consider the sentence:*

(3.20)	Les	idées	dorment	furieusement
	The	*ideas*	*sleep*	*furiously*

We are going to illustrate that for any model in the class of models we can verify that it is a model of the grammar. For doing so, we pick a complete model for the *sentence above, and we are going to show that under such a model all the constraints in the grammar* Γ *are satisfied (we will omit all the instances of rules that are trivially satisfied). We choose the model whose constituent structure can be represented as follows:*



We proceed gradually, bottom-up, starting from the surface level. Note that we do not describe here any specific parsing algorithm: in the perspective of Model Theory there is no notion of algorithm for building a model.

Each word in (3.20) corresponds to a constant w_i *where* i *is the position of the word in the sentence. We assume that all the words are in the lexicon; therefore each* w_i *is interpreted through* $\mathcal{I}$ *by the constituent in* $\mathcal{D}$ *of the lexical construction corresponding to the word in the sentence that* w_i *represents. At this stage, there are no constraints involved and each constituent is a model of the corresponding word.*
Each of these lexical constituents is now represented by a variable v_i *of position index* i *inherited from* w_i*. We assume that each word has a unique Part-Of-Speech (POS); since no constraints are involved at this stage either, like previously we end up with a new constituent for each POS, hence the following set of constituents:*

$$E = \{\mathrm{D}_1, \mathrm{N}_2, \mathrm{V}_3, \textsc{Adv}_4\}$$

In turn, each constituent in E *is represented by a variable, respectively* x_1 *to* x_4*. Let* r_5 *be a variable representing a hypothetical dominant constituent* R_5 *for* x_1 *and* x_2*. The construction of* R_5 *is yet unknown. We consider the assignment* $\mathcal{A}_5 = \langle \|r_5\| =$

$\mathrm{R}_5, \|x_1\| = \mathrm{D}_1, \|x_2\| = \mathrm{N}_2\rangle$. *The following constraints from* Γ *apply:*

$$\begin{aligned} &\vdash_{\textit{(P3.6)}}[\mathrm{D}](r_5) \wedge {}_{\textit{(P3.7)}}[\mathrm{D}](r_5) \wedge {}_{\textit{(P3.8)}}[\mathrm{N}](r_5) \wedge {}_{\textit{(P3.11)}}[\mathrm{D}, \mathrm{N}](r_5) \\ &\quad \wedge {}_{\textit{(P3.15)}}[\mathrm{N}, \mathrm{D}](r_5) \wedge {}_{\textit{(P3.18)}}[\mathrm{N}, \mathrm{D}, \mathrm{Desc}(x', y') : \begin{bmatrix} x'.\textsc{gend} & = & y'.\textsc{gend} \\ x'.\textsc{num} & = & y'.\textsc{num} \end{bmatrix}](r_5) \\ &\quad := \phi_5(r_5) \end{aligned} \tag{3.21}$$

When developed, the constraints are the following:

$$\begin{aligned} &\vdash (\Lambda x \,\mathrm{constituents}(r_5, x) \rightarrow \exists x_1 \,\mathrm{constituents}(r_5, x_1) \wedge x_1.\textsc{cat} = \mathrm{D}) \\ &\quad \wedge (\Lambda x_1 \Lambda y \,\mathrm{constituents}(r_5, x_1, y) \wedge x_1.\textsc{cat} = \mathrm{D} \wedge y.\textsc{cat} = \mathrm{D} \rightarrow x_1 = y) \\ &\quad \wedge (\Lambda x_2 \Lambda y \,\mathrm{constituents}(r_5, x_2, y) \wedge x_2.\textsc{cat} = \mathrm{N} \wedge y.\textsc{cat} = \mathrm{N} \rightarrow x_2 = y) \\ &\quad \wedge (\mathrm{constituents}(r_5, x_1, x_2) \wedge x_1.\textsc{cat} = \mathrm{D} \wedge x_2.\textsc{cat} = \mathrm{N} \rightarrow \mathrm{right}(x_1) < \mathrm{left}(x_2)) \\ &\quad \wedge (\Lambda x_2 \,\mathrm{constituents}(r_5, x_2) \wedge x_2.\textsc{cat} = \mathrm{N} \rightarrow \exists x_1 \,\mathrm{constituents}(r_5, x_1) \wedge x_1.\textsc{cat} = \mathrm{D}) \\ &\quad \wedge (\mathrm{constituents}(r_5, x_1, x_2) \wedge x_1.\textsc{cat} = \mathrm{D} \wedge x_2.\textsc{cat} = \mathrm{N} \rightarrow D[x_1/x', x_2/y']) \end{aligned} \tag{3.22}$$

The corresponding projection rules are the following:

$$\begin{aligned} &\vdash \Lambda r_5 \; r_5.\textsc{cat} = \mathrm{NP} \rightarrow \mathrm{Obl}\,(\mathrm{N} \vee \textsc{Pro}) \\ &\quad \wedge \Lambda r_5 \; r_5.\textsc{cat} = \mathrm{NP} \rightarrow \mathrm{D}! \\ &\quad \wedge \Lambda r_5 \; r_5.\textsc{cat} = \mathrm{NP} \rightarrow \mathrm{N}! \\ &\quad \wedge \Lambda r_5 \; r_5.\textsc{cat} = \mathrm{NP} \rightarrow \mathrm{D} \prec \mathrm{N} \\ &\quad \wedge \Lambda r_5 \; r_5.\textsc{cat} = \mathrm{NP} \rightarrow \mathrm{N} \Rightarrow \mathrm{D} \\ &\quad \wedge \Lambda r_5 \; r_5.\textsc{cat} = \mathrm{NP} \rightarrow \mathrm{N}\begin{bmatrix} \textsc{gend} \; \boxed{1} \\ \textsc{num} \; \boxed{2} \end{bmatrix} \leadsto \textsc{Det}\begin{bmatrix} \textsc{gend} \; \boxed{1} \\ \textsc{num} \; \boxed{2} \end{bmatrix} \\ &\quad := \psi_5 \end{aligned} \tag{3.23}$$

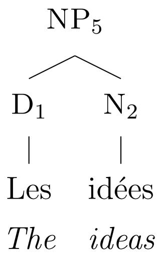
We then verify that:

$$\mathcal{I}, \mathcal{A}_5 \models {}_{\textit{(P3.6)}}(r_5) \wedge {}_{\textit{(P3.7)}}(r_5) \wedge {}_{\textit{(P3.8)}}(r_5) \wedge {}_{\textit{(P3.11)}}(r_5) \wedge {}_{\textit{(P3.15)}}(r_5) \wedge {}_{\textit{(P3.18)}}(r_5) \tag{3.24}$$

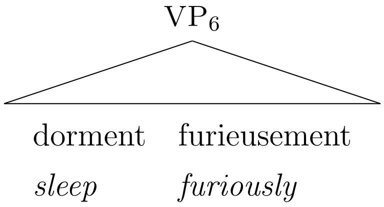
and:

$$\mathcal{I}, \mathcal{A}_5 \models \psi_5 \tag{3.25}$$

Therefore, all the constraints in the grammar are (strictly) satisfied under the model $\langle \mathcal{I}, \mathcal{A}_5 \rangle$*. The corresponding constituent* NP_5 *can be represented as follows:*



Following the same reasonning for the phrase dorment furieusement*, we now assume the existence of the consituent* VP_6*, represented as follows:*



Let x_5 *and* x_6 *be variables representing* NP_5 *and* VP_6 *respectively, and let* r_7 *be a variable representing a hypothetical dominant constituent* R_7 *for* x_5 *and* x_6*. We consider the assignment* $\mathcal{A}_7 = \langle \|r_7\| = \mathrm{R}_7, \|x_5\| = \mathrm{NP}_5, \|x_6\| = \mathrm{VP}_6 \rangle$*. The following* *constraints from* Γ *then apply:*

$$\begin{aligned}
&\vdash_{\textit{(P3.1)}}[\mathrm{VP}](r_7) \wedge {}_{\textit{(P3.2)}}[\mathrm{NP}](r_7) \wedge {}_{\textit{(P3.3)}}[\mathrm{VP}](r_7) \wedge {}_{\textit{(P3.4)}}[\mathrm{NP},\mathrm{VP}](r_7) \\
&\quad \wedge {}_{\textit{(P3.5)}}[\mathrm{NP},\mathrm{VP}](r_7) \\
&\quad := \phi(r_7)
\end{aligned} \tag{3.26}$$

$$\begin{aligned}
&\vdash \Lambda r_7\ r_7.\textsc{cat} = \mathrm{S} \rightarrow \triangle \mathrm{VP} \\
&\quad \wedge \Lambda r_7\ r_7.\textsc{cat} = \mathrm{S} \rightarrow \mathrm{NP}! \\
&\quad \wedge \Lambda r_7\ r_7.\textsc{cat} = \mathrm{S} \rightarrow \mathrm{VP}! \\
&\quad \wedge \Lambda r_7\ r_7.\textsc{cat} = \mathrm{S} \rightarrow \mathrm{NP} \prec \mathrm{VP} \\
&\quad \wedge \Lambda r_7\ r_7.\textsc{cat} = \mathrm{S} \rightarrow \mathrm{NP} \leadsto \mathrm{VP} \\
&\quad := \psi_7
\end{aligned} \tag{3.27}$$

We verify that:

$$\mathcal{I}, \mathcal{A}_7 \models \textit{(P3.1)} \wedge \textit{(P3.2)} \wedge \textit{(P3.3)} \wedge \textit{(P3.4)} \wedge \textit{(P3.5)} \tag{3.28}$$

$$\mathcal{I}, \mathcal{A}_7 \models \psi_7 \tag{3.29}$$

As a result, we can conclude that $\mathrm{R}_7.\textsc{cat} = \mathrm{S}$*; the constituent structure whose root is* S_7 *thus models the input. Note that Equation (3.28) informs on the characterisation (see Definition 3.14)* $\chi_{\mathcal{A}_7} = \langle \chi^+_{\mathcal{A}_7}, \chi^-_{\mathcal{A}_7} \rangle$ *of* S_7*: since* $\mathcal{A}_7$ *strictly satisfies all the constraints in the grammar, we have* $\chi^-_{\mathcal{A}_7} = \emptyset$ *and* $\chi^+_{\mathcal{A}_7} = \bigwedge\{\textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.4)}, \textit{(P3.5)}\}$.

Strict satisfaction allows models to be found for well-formed input sentences (*i.e.* for expressions), but it is not sufficient to model quasi-expressions, for which constraints fail. Therefore, in order to model ill-formed sentences we need the possibility of relaxing failed constraints. We define *loose satisfaction* for that purpose.

Definition 3.13 (Satisfaction (loose)) *Let* ϕ *be a formula of the form* $\phi = \bigwedge_{i \in \{\ 1 \ldots n\}} a_i$ *where for all* i*,* a_i *is an atomic formula. A model* $\mathcal{M}$ loosely satisfies ϕ *under the assignment* $\mathcal{A}$ *(denoted by* $\mathcal{M} \models\!\!\!\!\sim \phi$*) iff there exists* $I_k = \{i_1, i_2, \ldots, i_k\} \subseteq$

$\{1, \ldots, n\}$ *such that*

$$\mathcal{M} \models \bigwedge_{i \in \{1,\ldots,n\} \setminus I_k} a_i \wedge \bigwedge_{j \in I_k} \neg b_j$$

For short, we may sometimes write $\mathcal{A} \mathrel{|\!\!\simeq} \phi$*, assuming a default interpretation* $\mathcal{I}$*.*

Subsequently, a model which loosely satisfies a formula ϕ from $\mathcal{L}$ under an assignment $\mathcal{A}$ is said to be *loosely consistent*. Loose satisfaction is illustrated in Example 3.2.

Property 3.1 *Since* $\bigwedge_{j=0}^{m}(\neg b_j) \equiv \neg \bigvee_{j=0}^{m}(b_j)$ *the definition 3.13 is equivalent to:*

$$\mathcal{M} \mathrel{|\!\!\simeq} \phi \textit{ iff } \mathcal{M} \models \bigwedge_{i=0}^{n} a_i \textit{ and } \mathcal{M} \not\models \bigvee_{j=0}^{m}(b_j)$$

The formula ϕ is a constraint system, that is, a conjunction of atomic formulae. Under a given assignment a constraint system may partly hold and partly fail. The notion of loose satisfaction captures the possibility of implicitly relaxing the violated constraints as part of the satisfaction process. For convenience, the satisfied constraints $\bigwedge_{i \in \{1,\ldots,n\} \setminus I_k} a_i$ are referred to as $\chi^+_{\mathcal{A}}$, while the violated constraints $\bigwedge_{j \in I_k} \neg b_j$ are referred to as $\chi^-_{\mathcal{A}}$.

Definition 3.14 (Characterisation) *In PG, the pair* $\langle \chi^+_{\mathcal{A}}, \chi^-_{\mathcal{A}} \rangle$ *is called the* characterisation *of* $\mathcal{A}$*, denoted by* $\chi_{\mathcal{A}}$*.*

Example 3.2 (Loose consistency and ill-formedness) *Consider the following ill-formed sentence in French (which is missing a determiner between positions 3 and 4):*

Le	*juge*	*octroie*	*bref*	*entretien*	*à*	*ce*	*plaignant.*
The	judge	grants	brief	interview	to	this	plaintiff.

We focus on building a model for the ill-formed NP (framed in Figure 3.1). Suppose the grammar Γ*, and the variables* x *and* y *representing respectively* AP_7 *and* N_9*. We consider* R_6 *a hypothetical dominant constituent for* x *and* y*, with* r *the variable representing* R_6*. Let* $\mathcal{A}$ *be the assignment such that* $\mathcal{A} = \langle \|r\| = \mathrm{R}_6, \mathrm{AP}_7, \mathrm{N}_9 \rangle$*. The* *following constraints in* Γ *apply:*

$$\begin{aligned}&\vdash \textit{(P3.6)}(r) \wedge \textit{(P3.16)}(r) \wedge \textit{(P3.15)}(r) \wedge \textit{(P3.17)}(r)\\ &\quad := \phi(r)\end{aligned} \tag{3.30}$$

$$\begin{aligned}&\vdash \Lambda r\ r.\textsc{cat} = \mathrm{NP} \rightarrow \phi(r)\\ &\quad := \psi\end{aligned} \tag{3.31}$$

We verify that:

$$\begin{aligned}&\mathcal{A} \models \textit{(P3.6)} \wedge \textit{(P3.16)} \wedge \textit{(P3.17)}\\ &\qquad := \chi^{+}_{\mathcal{A}}\end{aligned} \tag{3.32}$$

$$\begin{aligned}&\mathcal{A} \not\models \textit{(P3.15)}\\ &\qquad := \chi^{-}_{\mathcal{A}}\end{aligned} \tag{3.33}$$

Therefore, by loose satisfaction we have $\mathcal{A} \mathrel{|}\!\backsimeq \phi$*, and* $\mathcal{A} \mathrel{|}\!\backsimeq \psi$*. By* ψ *we can project the construction* NP *for* R_6*, which gives the constituent* NP_6*. The constituent* S_{15} *can then be reached by strict satisfaction (see Figure 3.1).*

Note that *strict satisfaction* is a special case of *loose satisfaction* where $\chi^{-}_{\mathcal{A}}$ is empty. The case where $\chi^{+}_{\mathcal{A}} = \emptyset$ and $\chi^{-}_{\mathcal{A}} \neq \emptyset$ is also potentially controversial as it raises the question of where to draw the line: if none of the atomic formulae in ϕ are true then is ϕ still *satisfied*, even loosely? We leave it up to applications and implementations to restrict the definition(s) further. A typical restriction consists of fixing a minimum number of strictly satisfied constraints, usually one. Should the desired syntactic structure be more specific, such as a head-driven phrase structure, another restriction can consist of fixing the type of constraint which must hold, in that case the presence of a head. In Chapter 4, which is dedicated to the presentation of a parsing strategy based on Ξ, we state five different assumptions that we make in order to further specify the type of constituent structure we are interested in for objects in $\mathcal{D}$ (see §4.3.2).

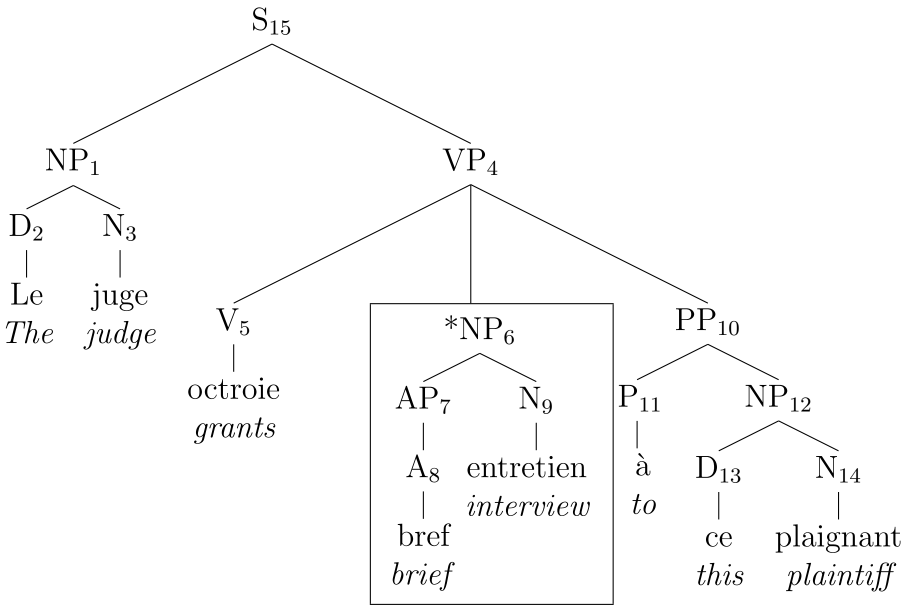


Figure 3.1: Loosely consistent constituent structure for an ill-formed sentence, which is missing a Determiner in NP

3.3 Properties of Ξ and Discussion

3.3.1 A Discriminant for Multiple Loose Models

Loose consistency comes at a cost: the multiplicity of candidate models. A possible discriminant is the optimality of a cost to be associated with each candidate model. Such a solution was adopted by VanRullen (2005), with the use of the *density of satisfaction* of a constituent structure. Translated in Ξ the density of satisfaction corresponds to the proportion of satisfied constraints of a model. For $\mathcal{M} \mathrel{|}\!\!\simeq \phi$, with $\mathcal{M} \models \chi^+_{\mathcal{M}}, \mathcal{M} \not\models \chi^-_{\mathcal{M}}$, the score of $\mathcal{M}$ is calculated by $\mathrm{card}(\chi^+_{\mathcal{M}}) / \mathrm{card}(\phi)$ where $\mathrm{card}(x)$ is the number of atomic formulae in x. The rationale is quite intuitive: the higher the ratio of satisfied conditions of well-formedness the better.

Of course, it might be desirable to fine-tune that cost in order to take into account differences among constraints and constraint types in the role they play and the importance they have with respect to grammaticality. For that reason, different constraint-based frameworks for natural language processing (Schröder, 2002; Keller, 2006), including PG, assign weights to the constraints. Meanwhile, it is also argued (Aarts, 2007) that since the procedures for modelling the weighting of syntactic properties are controversial from a linguistic point of view, a computational model for degrees of grammaticality should weight all constraints equally. Yet we do not completely agree on that the controversy around procedures make the very use of weights controversial, and we have chosen to adopt constraint weights. In §4.4.4 we present the solution we have implemented.

3.3.2 Constituent Structure

Because the constituency relationship is hierarchical we illustrate it as a tree. But it must be emphasised that such a tree is only remotely related to the notion of tree as used in generative grammar. As we have just seen through this chapter the tree structure as such, usually formalised in terms of edges and vertices, does not play any particular role in the logical framework. Besides, an isolated constituent tree does not describe all syntactic aspects of the constituent(s)—*i.e.* of the input string it describes; the complete information about the syntactic properties of a string is contained in the satisfied constraint system which yielded the constituent hierarchy. It is a strength of Property Grammars to provide, with the notion of *characterisation*, a more informative—because more comprehensive—representation of that syntactic knowledge.

This being said, it may be helpful for a better understanding to draw an analogy between a production rule from generative grammar and an inference rule in Ξ.

Let us consider the constraint $\phi(r)$ paired up with the projection rule $\psi : \Lambda r\ r.\textsc{cat} = \mathrm{L} \rightarrow \phi(r)$, and the assignment $\mathcal{A} = \langle \mathrm{L}, \mathrm{R}_1, \mathrm{R}_2, \ldots, \mathrm{R}_n \rangle$. The generation—in the sense of generative grammar—of the constituent from the type L would require rewriting using the following production rule:

$$\mathrm{L} \longrightarrow \mathrm{R}_1\ \mathrm{R}_2\ \ldots\ \mathrm{R}_n \tag{3.34}$$

Yet, as we said previously, the comparison between the two remains only superficial and incomplete, since the rule (3.34) does not capture either the possibility of loose satisfaction or the syntactic information contained in the characterisation about relationships among R_i's. Another difference stands in that the R_i's in the PG version are values from the domain $\mathcal{D}$ (semantic level), whereas the R_i's in the production rule belong to the language (syntactic level).

3.3.3 Grammaticality

As far as representing natural language is concerned, grammaticality comes in significantly different flavours. In Chapter 2 we have identified three main notions, which co-exist in the literature: a *generative-theoretic* notion, where a sentence is either grammatical or not; an *optimality-theoretic* notion, where a syntactic structure is grammatical, possibly to a certain degree, if it is the optimal one among a set of candidates; and a *model-theoretic* notion, where a sentence is either grammatical or ungrammatical to a certain degree. Arguably there exists a fourth notion, *constructional* grammaticality, where a sentence is grammatical or ungrammatical with respect to a class of expressions—a *construction*, possibly to a certain degree, whether of grammaticality or ungrammaticality. We consider here that constructional grammaticality is a special case of model-theoretic grammaticality.

In short, the generative-theoretic grammaticality is strictly binary, and nothing can be said about ungrammatical sentences. Quasi-expression and degree of ungrammaticality are meaningless notions in that case. The optimality-theoretic grammaticality takes its name from Optimality Theory (OT) (Prince and Smolensky, 1993), but is found, for what we are interested in here, in other approaches to natural language processing (Schröder, 2002; Keller, 2006). OT does not discriminate between grammatical and ungrammatical sentences. Instead, it discriminates the optimal candidate structure for an input sentence according to preference criteria, pre-supposing the existence of a set of candidates.

The Model-Theoretic grammaticality, as we formalise it, combines the sharp distinction between grammatical and ungrammatical sentences, the optimality of the syntactic structure, and the classification of utterances in constructions. This combination is made possible by linking the grammaticality of a sentence to the consistency of its structure, with respect to a specific construction, and by allowing constituents to be paraconsistent (*i.e.* loosely consistent) in order to model ungrammatical input utterances. Furthermore, we will see in Chapter 5 that it is also possible to model a gradient of ungrammaticality in rating an utterance on the basis of the information contained in its paraconsistent model.

3.3.4 Monotonicity

In mathematics, a function is said *monotonic* when it is steadily increasing or steadily decreasing when the variable is incremented. Similarly, a constraint is said to be *monotonic* when it is steadily strictly satisfied or steadily violated when the assignment is incremented—an increment being a constituent. Dahl and Blache (2004) give the following definition:

Definition 3.15 (Monotonicity, Dahl and Blache, 2004) *Let `XP` be a phrase into which we are considering incorporating a category `Cat`. Let us designate by `XP+Cat` the new constituent (of type `XP`) formed by incorporating `Cat` into `XP`. A property*[5] *P is said to be* success-monotonic *(*failure-monotonic*) if all P properties that hold (fail) in `XP` for $\mathcal{A}$ also hold (fail) in `XP+Cat`.*

Let us try and adapt that definition in Ξ. The phrase `XP` is a constituent, whose sub-constituents we denote by $\{x_1, \ldots, x_n\}$. Let us make an assignment out of these sub-constituents along with `XP`, and denote it by $\mathcal{A} = \langle \texttt{XP}, x_1, \ldots, x_n \rangle$. Let c be the constituent of construction `Cat` to be incorporated in the new constituent `XP+Cat`. The meaning of "incorporating a category `Cat` into `XP+Cat`" translates in Ξ into `XP+Cat` being modelled by the assignment $\mathcal{A}' = \langle \texttt{XP}, x_1, \ldots, x_n, c \rangle$. The monotonicity of a constraint P is then defined in Ξ by:

$$P \text{ is success-monotonic iff if } \mathcal{A} \models P \text{ then } \mathcal{A}' \models P \tag{3.35}$$

$$P \text{ is failure-monotonic iff if } \mathcal{A} \not\models P \text{ then } \mathcal{A}' \not\models P \tag{3.36}$$

[5] The term *property* is taken here in the sense of PG. In that case, it should be read as *constraint.*

The monotonicity of a constraint is illustrated[6] in Example 3.3.

Example 3.3 *The Linearity constraints are both success–monotonic and failure–monotonic, while the Uniqueness constraints are failure-monotonic only.*

Consider the constraints:

$$\phi_1 = \text{lin}[\textsc{Det}, \text{N}](r) \tag{3.37}$$

$$\phi_2 = \text{uniq}[\text{V}](r) \tag{3.38}$$

and the following assignments:

$$\mathcal{A}_1 = \langle R_1, \text{D}_{11}, \text{A}_{12}, \text{N}_{13} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{D}_{11}, \textit{ and } \|y\|_{\mathcal{A}} = \text{N}_{13} \tag{3.39}$$

$$\mathcal{A}'_1 = \langle R'_1, \text{D}_{11}, \text{A}_{12}, \text{N}_{13}, \text{V}_{14} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{D}_{11}, \textit{ and } \|y\|_{\mathcal{A}} = \text{N}_{13} \tag{3.40}$$

$$\mathcal{A}_2 = \langle R_2, \text{N}_{21}, \text{D}_{22} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{D}_{22}, \textit{ and } \|y\|_{\mathcal{A}} = \text{N}_{21} \tag{3.41}$$

$$\mathcal{A}'_2 = \langle R'_2, \text{N}_{21}, \text{D}_{22}, \textsc{Pro}_{23} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{D}_{22}, \textit{ and } \|y\|_{\mathcal{A}} = \text{N}_{21} \tag{3.42}$$

$$\mathcal{A}_3 = \langle R_3, \text{V}_{31} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{V}_{31} \tag{3.43}$$

$$\mathcal{A}'_3 = \langle R'_3, \text{V}_{31}, \text{V}_{32} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{V}_{31} \tag{3.44}$$

$$\mathcal{A}''_3 = \langle R''_3, \text{V}_{31}, \text{V}_{32}, \text{N}_{33} \rangle, \textit{ with } \|x\|_{\mathcal{A}} = \text{V}_{31} \tag{3.45}$$

For Linearity we have the following:

$$\mathcal{A}_1 \models \phi_1 \tag{3.46}$$

$$\mathcal{A}'_1 \models \phi_1 \tag{3.47}$$

[6] Presenting proofs of the monotonicity properties of a constraint would be quite interesting, as would be a logical/epistemological decomposition of constraints into more standard operators of classical logic that have these properties. However, the only proofs we can think of would require a more detailed axiomatisation, which should allow—for the essentials—for reasonning with sets of variables; it would also require a number of assumptions to be axiomatised, in particular with regard to the contiguity of members in a set (otherwise Linearity must be re-defined). We might be wrong of course, but we strongly suspect that providing these axiomatisations in FOL is not quite straightforward.

and:

$$\mathcal{A}_2 \not\models \phi_1 \tag{3.48}$$

$$\mathcal{A}'_2 \not\models \phi_1 \tag{3.49}$$

While for Uniqueness *we have:*

$$\mathcal{A}_3 \models \phi_2 \tag{3.50}$$

$$\mathcal{A}'_3 \not\models \phi_2 \tag{3.51}$$

$$\mathcal{A}''_3 \not\models \phi_2 \tag{3.52}$$

$$\tag{3.53}$$

The notion of monotonicity can actually be generalised to constraint types, since the monotonicity of a constraint originates in its type's semantics. Dahl and Blache identify three categories of properties (*i.e.* constraints) according to their monotonicity:

- the **selection properties**, which are both success- and failure-monotonic (*Linearity*, *Dependency*);
- the **filtering properties**, which are failure- but not success-monotonic (*Uniqueness*, *Exclusion*);
- the **recoverable properties**, which are success- but not failure-monotonic (*Requirement*, *Obligation*).

A constraint (type) is said to be *monotonic* when it is both success-monotonic and failure-monotonic. We also use the term *non-monotonic* to refer to a constraint or a constraint type in general, which is exclusively either success- or failure-monotonic, and when being more specific is irrelevant in context.

Dahl and Blache report that their parsing strategy only takes into account the monotonic constraints (*i.e.* the selection properties) for the projection of a construction. Likewise in VanRullen (2005), where the non-monotonic constraints are called *lacunar properties*. VanRullen implements a *Principle of non-projection on lacunar properties*. In doing so, these strategies overlook an important feature and motivation of PG, whereby a constituent structure is not head-driven, nor is it driven by any type of constraint. In particular, the structure is meant not to be head-driven. In making judgements about the construction of a dominant constituent relying on the monotonic constraints, only these strategies only partially implements that feature of PG.

3.3.5 Some Related Works

Availability In their formalisation of PG, VanRullen, Guénot, and Bellengier (2003; 2005) (VRetAl) define the *availability* of a logical term, and of a constraint. The availability is motivated by a procedural rationale, in order to capture that in the context of an incremental parsing strategy an object from the domain might *become* available, as an assignment is dynamically incremented. Our formalisation differs in that such a dynamic modification of an assignment is not directly accounted for. Meanwhile, it can be emulated in seeing the assignment after each increment as a brand new assignment. An axiomatic could be defined in order to maintain the link between the two assignments, before and after the increment.

In this respect, we would like to draw attention on the similarity between VRetAl's availability and the Assignment function as we defined it above (see Definition 3.4). In mapping variables to values in $\mathcal{D}$ the function $\mathcal{A}$ only applies to the *theory* concerned, that is, to the set of formulae of the variables concerned. Thus a constraint is available, in the sense of VRetAl, if its variables are mapped to values in $\mathcal{D}$ by $\mathcal{A}$. The requirement in VRetAl for a constraint to be available in order to be evaluable is captured in Ξ by the fact that a model can only (loosely) satisfy a theory *under an assignment* $\mathcal{A}$.

Capacity of Constraint Type For VanRullen,

> La capacité [d'un type de propriété] indique le nombre de fois qu'une même propriété peut être satisfaite pour une même construction. (VanRullen,

2005, p. 68)
The capacity [of a property type] indicates the number of times that the same property may be satisfied for the same construction.

The capacity of a constraint type is expected to be provided with its specification. We believe that the definition of the capacity is essentially motivated by specific algorithmic requirements. A case at stake is, for instance, the one of Linearity: for the same assignment one instance of Linearity may hold for a pair $\{x, y\}$ of constituents, and fail for another. Consider, for example, the following sentence along with the words' POS:

(3.54)	The	day	the	student	graduates
	D_1	N_2	D_3	N_4	V_5

Under the assignment $\mathcal{A} = \langle \mathrm{R}, \mathrm{D}_1, \mathrm{N}_2, \mathrm{D}_3 \rangle$ such that
$\|r\|_{\mathcal{A}} = \mathrm{R}$, $\|x_1\|_{\mathcal{A}} = \mathrm{D}_1$, $\|y_2\|_{\mathcal{A}} = \mathrm{N}_2$, $\|x_3\|_{\mathcal{A}} = \mathrm{D}_3$ the Linearity constraint is satisfied for the variables $\{x_1, y_2\}$, but is violated for $\{x_3, y_2\}$. The idea of VRetAl's capacity is to differentiate these two instances of Linearity, and to enable the specification of constraint types, which constrain the number of these instances. Linearity, for example, is specified by VRetAl with a capacity that takes its value in $(0; +\infty)$. However, in practice the capacities of the Linearity constraints in all the different grammars developed by VRetAL are never valued more precisely. The reason for that is that each constraint of Linearity in the grammar is specified for all pairs of variables, under a specific assignment. The need for making the distinction between instances can be justified from an implementation point of view, for instance for maintaining constraint counts, but from a logical point of view the constraint applies to all variables in the assignment. Transposed in the formal context that we are using, the capacity of a constraint type would correspond to a valuation of the quantifiers involved in the type's specification. This way, it would be possible to specify a constraint type which constrains the number of satisfied instances through valued quantifiers. And if reasonning with specific instances is needed, then one can define the constraint type $\mathrm{local\_lin}[\mathrm{C}_1, \mathrm{C}_2](r, x, y)$ such that $\mathrm{linearity}[\mathrm{C}_1, \mathrm{C}_2](r) \Leftrightarrow \Lambda x \Lambda y \, \mathrm{local\_lin}[\mathrm{C}_1, \mathrm{C}_2](r, x, y)$. In practice however, the constraint types currently in use in PG only require the quantifiers to be valued in two different ways:

- a capacity of 1 corresponds to the existential quantifier;
- a capacity of unspecified value in $(0; +\infty)$ corresponds to the quantifier Λ.

Therefore, and although the use of valued quantifiers might look theoretically interesting, it seems, so far, not to be justified in practice. Moreover, since its introduction would require using a higher-order logic we prefer the solution adopted in Ξ.

Robustness The property of robustness is usually one of a process or program, and refers to the ability to handle unexpected or deviant input without failure. Although the topic of the present chapter does not include any processing aspect, it does provide important formal tools to address the question of the robustness of a parsing process, which ought to be emphasised.

Achieving robustness is necessarily purpose-driven and we certainly do not claim to answer the problem as a whole. Meanwhile, we do claim that the model-theoretic framework we provide serves the (theoretical) purpose of generating a full deep structure for unrestricted input. Irrespective of any processing considerations, we observe that through loose satisfaction a constituent structure may be a model for a deviant input even though it fails to satisfy every specification.

Note, as well, that the observation of constraint failures is part of the regular satisfaction process, and does not require more information than what is contained in the grammar. This aspect of PG makes the framework differ significantly from other approaches which aim to handle deviant language. The mechanisms of *constraint relaxation* implemented in most knowledge-based robust parsers rely on an explicit set of rules for relaxation, such as error rules or mal-rules (Weischedel and Sondheimer, 1983; Carbonell and Hayes, 1983; Mellish, 1989; Douglas and Dale, 1992; McCoy, Pennington, and Suri, 1996; Bender et al., 2004; Foster and Vogel, 2004). Applying such mal-rules typically involves a post-processing (or *post-mortem*) step extending conventional parsing and depending on the strategy used, in order to recover from failures. Such an error grammar[7] may play different roles, whether to identify and/or correct ill-formedness accordingly, but conceptually it always aims to specify errors (or

[7] The term was coined by Foster and Vogel (2004).

classes of errors). In doing so, it enlarges the set of sentences covered by the grammar to include ill–formed ones—that is, it expands the language under investigation to include an additional and limited (besides the cases of infinite strings) set of sentences otherwise considered ill-formed. The major drawback of such an approach is still to only cover the subset of language being specified, ignoring the remainder.

The approach taken in PG is significantly different. It offers a general framework for handling language as a whole, regardless of well-formedness, and allows parsing any input string. Therefore, the problem of remaining uncovered utterances becomes theoretically irrelevant[8]. Interestingly however, the two approaches are not completely inconsistent. Nothing in PG prevents specifying in the grammar syntactic properties for classes of ill-formed constructions. Such a policy may improve coverage and accuracy of the parser—though we must leave that question open, since we opted for a different grammar writing strategy.

The next chapter (Chapter 4) is dedicated to describing the—robust—parsing strategy we have developed.

3.4 Conclusion

In this chapter we have defined a logical system for Model-Theoretic Syntax based on First-Order Predicate Logic with a Model-theoretic semantics, in which the paradigm of Property Grammars (PG) defined in Blache (2001, 2005) can be formulated. The contribution of our formalisation, compared with the existing formalisations of PG, is many-fold. First, it makes an explicit link between PG and the family of MTS frameworks. Second, in such a model-theoretic perspective, our formalisation abstracts away from any notion of algorithm for building a model. Third, it captures the projection mechanism in logical terms through the definition of *projection rules*, similar to type judgements. Fourth, it defines *loose satisfaction* as a non-classical logical consequence, in order for consistent and parconsistent models to loosely satisfy the constraints in the grammar, by enabling the relaxation of the violated constraints.

[8] Practically, however, we acknowledge that performance-related limitations remain, which we present and discuss in Chapters 4 and 5.

The well-formedness of a sentence is thus captured by the consistency of the model, while ill-formedness is captured by loose consistency. Formulations were provided for all the different standard constraint types in use in PG.

Chapter 4

Loose Constraint-based Parsing

One of the main difficulties with configuration problem solving lies in the representation of the domain knowledge because many different aspects, such as taxonomy, topology, constraints, resource balancing, component generation, etc. have to be captured in a single model.
(Mailharro, 1998, p. 383)

4.1 Introduction

We have seen previously that, in general terms, a model for gradience applies to an input item which is properly classified, as well as characterised in terms of (met and broken) features. Such classification involves a judgement of acceptability based on whether the item's features meet those of the class and how important these features are for the class. Before being able to model this acceptability judgement the input item must, therefore, be correctly characterised, in terms of features, and correctly classified. The aim, in this chapter, is to develop a tool, which precisely characterises and classifies an input item, and thus provides all the input material required by the computational model. As far as the constituent structure is concerned, by *classifying* a string we mean providing a full parse tree for it, which is rooted by the class the string belongs to—in our case a construction. Therefore the outcome (of the classification task) must always be a full parse tree, whether the input item is deviant or not.

Although the requirements we have sound very much like those for a robust parser the usual strategies implemented in robust parsing fail to meet our expectations in terms of coverage and nature of the outcome and are, therefore, not well-suited to a theoretical generalisation of a model of syntactic gradience. Indeed, all the parsers whose outcome may exclusively be a partial parse for deviant language are ruled out, since the input string could not be considered as uniformly classified into one construction and the model for gradience would not apply to it. The stochastic approaches to robust parsing, on the other hand, would, intuitively, probably be sufficient, but they require to be trained on appropriately annotated corpora. Besides the practical issue of gathering such corpora and/or designing a large-coverage language model, it is not clear how such approaches, where errors are only represented implicitly, could produce an output which could be used in predicting gradience[1]. A possible approach uses machine learning techniques, as investigated by Mutton et al. (2006), who developed an evaluation metric to estimate text fluency. As for the knowledge-based approaches, whether based on mal-rules for grammar checking (Bender et al., 2004), error grammar (Foster and Vogel, 2004), (traditional) constraint relaxation (Douglas and Dale, 1992), or any other recovery process they all present a similar drawback in that they require user-defined specific modules to be applied on top of a main grammar in order to deal with ill-formedness. Such modules are still limited in coverage. Therefore a model of gradience based on them would also be limited to the same coverage and a theoretical generalisation would still not be possible. Parsing strategies based on PG also exist which are an exception to that limitation in coverage, but in their case they are still not suitable because their outcome can not be proven optimal. Indeed, as we already emphasised it especially in Chapter 2, §2.5.4, incremental strategies come along with an inadequate pruning of the search space.

[1] Keller's LOT, to some extent, should be seen as an exception, though the approach does not allow for discriminating deviant from non-deviant language. This point is discussed in more details in §2.3.2.

Relying on Existing Approaches

SeedParser

VanRullen's implementation of a PG parser (SeedParser) does not present such a limitation in terms of coverage since it does handle any input string, whether well- or ill-formed, for which a solution parse is provided. There are nonetheless different aspects in VanRullen's work that prevent us from relying on the SeedParser in our own study.

Bearing in mind that the parser's output will be used later on for rating the input string we require the parser in use to provide us with a syntactic parse and a characterisation for a full sentence that can be proven of optimal merit. As far as the SeedParser is concerned it is not clear to us whether it can be proven:

1. that a full syntactic parse will always be found, if it exists; and
2. that the result parse is of optimal merit.

Typically, it is unclear whether inapropriate pruning of the search space could result from the incremental parsing strategy as adopted. By *inapropriate pruning* we mean, for instance, that the decision process whereby the boundaries of a constituent are decided might ignore solutions for grouping differently the same set of sub-constituents, without any means to control whether one (or more) of these alternative(s) would be a better choice for the overall merit of the full input sentence. The delaying of the evaluation of certain non-monotonic constraints also potentially influences the decision process for categorising a constituent in a way that may impact the optimality of the chosen solution.

Therefore the parsing strategy implemented in the SeedParser does not adequately suit our purpose.

Constraint Programming Approaches

Following Dahl and Blache (2004) and Morawietz and Blache (2002) we considered relying on Constraint Logic Programming (CLP)—namely Constraint Handling Rules

(CHR) (Fruehwirth, 1995)—for implementing a parser for PG. In fact, we did implement a couple of experimental parsers based on CHR prior to implementing the version we present in this chapter, but in each case we were confronted to different issues which led us to abandon the Constraint Programming paradigm. The first issue at stake comes from that we do not control the Constraint Satisfaction process which is handed over to the Constraint Soving engine. Therefore the process of Loose Constraint Satisfaction, controlled by the merit of a constituent, could not be implemented. Could not be implemented either different filters which would aim either to prune the search space on linguistic ground or to skip cases that be known to fail.

The second issue comes from a well-known drawback of Logic Programming and concerns the practical difficulty one is faced with when dealing with complex data structures.

It is not entirely clear though, whether these impediments are all to be put on the account of CHR and CLP, or whether some of them are more linked to the specific implementations of CHR that we used. In a attempt to address these two main problems using CLP we implemented a parsing algorithm which made use of a Java implementation of CHR. Both the Java Constraint Kit (JaCK)[2] and the K.U.Leuven Java CHR System (JCHR) (Weert, Schrijvers, and Demoen, 2005) were experimented—the only two APIs available, to the best of our knowledge, at the time of programming. As far as representing complex data structures is concerned the solution is indeed very elegant and satisfactory. As for controlling part of the satisfaction process it is unfortunately unclear, at this stage, to what extent it is possible. Several of the problems we encountered were likely to come from that either of these APIs were still in the early stage of their development (JaCK was actually already no longer maintained), and while they did provide a convenient interface between CHR and Object Oriented Programming features they were still limited, for instance in terms of features available in order to search the solution space. Quite obviously in our context such search facilities are a strong requirement in order to recover and walk through the characterisation of constituents. It is also not clear to us whether it be possible and how to

[2] It seems that no publications are related to JaCK, though a documented website exists and can be found at: `http://www.pms.ifi.lmu.de/software/jack/` (online on the 27 August 2008)

implement a process of loose constraint satisfaction.

For all these reasons we developed an algorithm for Loose Satisfaction Chart Parsing (LSCP) in order to meet our requirements and we show in the present chapter that it is correct by construction. It is correct in that the solution it outputs for a—possibly deviant—input string is a full parse tree and is proven to always optimise a merit function. More specifically, the input item being a string, classifying it is interpreted as finding an optimal syntactic structure for it, while characterising it is interpreted in the standard sense from PG. The result syntactic structure is a constituent structure. Provided the model-theoretic framework we presented in the previous chapter, finding an optimal structure is equivalent to finding an optimal model for the input, which loosely satisfies inference rules from the grammar.

It is important to emphasise that our goal in developing an LSCP algorithm is primarily theoretical: we aim to implement a parser whose output is proven optimal according to a merit function, so that we know—in Chapter 5—that we are using the best possible syntactic parse and characterisation for rating an input sentence. This means that it is important to prove that the optimality of the output over the entire search space. The use of heuristics should not prevent that goal, and is, therefore, quite limited at this stage. In the conclusion section (§4.10) we envisage different possibilities of optimisation in further works.

The chapter is organised around demonstrating the correctness of the algorithm. As a preamble, we introduce and discuss in §4.2 the choices we made in terms of knowledge representation. Then we proceed by stating, in §4.3, the exact problem we are faced with, and specifying the expected solution; following is §4.4, where we describe the algorithmic solution we have implemented for LSCP, and we show how to build a correct solution through a step-by-step procedure. §4.6 demonstrates the theoretical complexity of LSCP, and §4.4.4 presents and discusses the merit function we are using for choosing the optimal structure. And finally §4.8 focuses on the heuristic we have implemented in order to prune the search space and drop the practical complexity of LSCP, while §4.9 presents an evaluation of the parser's outcome.

4.2 Presenting And Representing Syntactic Knowledge

We have already started in §2.5.3 discussing the particular way of representing syntactic knowledge in PG. We complement here the discussion in precising important aspects of (syntactic) knowledge representation (KR) and our interpretation of them.

A powerful feature of PG regarding KR stands in the distant connection it introduces between syntactic structure and procedural mechanisms. Such a distance between knowledge structure and procedure offers a high flexibility concerning the type of representation in use and the interpretation associated to it.

In §3.2.2 we described a *constituent* as a composite object showing a hierarchical structure of constituents, and we presented a constituent as a basic object from the domain of discourse. We now detail the representation we have adopted for a constituent.

The component-based architecture proposed in Mittal and Frayman (1989) (see §2.5.7) as a generic representation for configuration tasks captures well all the different views exposed in the literature. We have adopted their suggestion, and adapted it to natural language processing. We see a *constituent* as a *component*, where *component* is taken in the sense defined in Mittal and Frayman (1989). Components are organised in a hierarchical structure, where the hierarchy relationship is a *dominance* relationship. Siblings, in this hierarchy, are organised according to a left-to-right linear order, and are connected among each others through *typed relationships*. These typed relationships among siblings are seen as *properties* of their *dominant constituent*. A typed relationship is represented as a constraint, as defined in Chapter 3.

Therefore, the traditional tree representation of a constituent structure only represents part of the syntactic knowledge contained in a constituent. A *syntactic tree* is merely a representation of both the linear order and dominance relationships among constituents, but which does not account for typed relationships mentioned above. A complete representation of all the relationships within one structure is the *multi-graph* one adopted in VanRullen (2005). Although exhaustive and therefore extremely powerful, the multi-graph representation also lacks an intuitive reading and can quickly turn quite difficult to grasp as its complexity increases. For that reason we choose to maintain in two different structures the typed relationships among siblings on one hand, and the constituency (dominance and linear order relationships) on the other hand. The former is maintained in a constituent's *characterisation*, as defined in Definition 3.14, while the latter is maintained in a *constituent structure*, as defined below in definition 4.1.

The outcome of the parsing process—which makes the topic of the present chapter—is a constituent, presenting a component architecture. It is presented as a combination of both its *constituent structure* and its *characterisation*.

4.3 Problem Specification

4.3.1 Problem Statement

The goal we have is, given an input string of unrestricted language, to build and characterise an optimal spanning constituent for it. A constituent is optimal if it has a maximum merit. The merit of a constituent, detailed in §4.4.4, is computed on the basis of its characterisation.

Put differently: given a grammar formulated in Ξ, and an input string annotated with atomic constituents (POS-tags), find a model for the input string which loosely satisfies the grammar. The model must be of maximum merit.

In the following section we assert strict principles to further specify and constrain, next to the grammar, what a legal constituent structure is.

4.3.2 Outcome

The expected outcome of the parsing process is a characterisation and a*constituent structure*, as defined below.

Definition 4.1 (Constituent Structure) *A* Constituent Structure *is a tree representation of the hierarchical internal structure of a constituent, which accounts for the relationships of dominance and linear order among constituents. It is represented as* *a finite labelled connected directed acyclic graph (DAG), satisfying Assumption 1 to 5 below, where the nodes are constituents from the domain of discourse $\mathcal{D}$ (see §3.2.2 in Chapter 3). It is directed according to the relationship of dominance.*

For short, we may also sometimes refer to a constituent structure as a *parse tree.* Note that the parse is supposed *finite.* Therefore, and following Pullum and Scholz (2001), this definition prevents infinite recursion.

Assumption 1 (Minimum Satisfaction) *Every constituent in a parse tree strictly satisfies at least one constraint.*

Assumption 1 is a restriction on the notion of loose satisfaction, as defined in definition 3.13. As we briefly mentioned it in §3.2.5, definition 3.13 raises the problem of an empty positive characterisation (χ^+, the set of satisfied constraints). We restrict that definition further by forcing χ^+ to be non-empty, *i.e.* to contain at least one satisfied constraint.

However, we put no restriction on the type of constraint that must be strictly satisfied. Especially, it does not have to be an *obligation* one, which means that the solution structure is not necessarily head-driven.

Assumption 2 (Spanning Parse) *The solution parse must span the entire input string. That is, every word from the input string must be a terminal in the solution parse.*

In case a full parse can not be found using the constructions specified in the grammar, we make use of a wildcard construction (called *Star*). This construction is not specified by any constraint. Its purpose is merely to span withing one root construction a forest of partial parses.

Assumption 3 (Distinct Constituents) *A constituent may contain no pairwise intersecting constituents.*

That is, within a constituent there may be no overlapping sub-constituents.

Assumption 4 (Constituent Arity) *A constituent from a POS-tag construction has an arity of 1 (one); otherwise, a constituent has a minimum arity of 2 (two), except when it is an immediate dominant of a POS-tag constituent, in which case the dominant constituent may have an arity of 1.*

Assumption 4 aims to prevent the risk of infinite recursion in ruling out constituents made of a single immediate sub-constituent. In forcing constituents to be made up of at least two immediate sub-constituents, the possibility of recursion is necessarily limited to the size of the input string. An exception is made for constituents immediately constituted of a single POS-tag, since it is commonly accepted that, for instance, a single Noun may constitute a Noun Phrase, or a single Adjective may constitute an Adjective Phrase.

Assumption 5 (Continuous Constituent) *Every constituent in a parse is continuous, that is, all its constituents are next to each others.*

Assumption 5 rules out gaps, and cross-serial dependencies. We acknowledge their necessity for French or English for phenomena such as extraposition, but these cases involve important issues of grammar development, and fixing them goes beyond the scope of our work. The main problem involves the semantics of some of the pre-defined constraint types, which would require to be re-defined. Linearity, for instance, would need to account for the order between two crossing-over phrases, which is not the case in the current definition (see Definition 3.5).

4.4 Algorithmic Solution

In order to address the problem we have just specified, we introduce the Loose Satisfaction Chart Parser (LSCP), and we present here its algorithm. We named our implementation of it *Numbat*[3]. In order to ease its understanding, we proceed by successive presentations of the entire process, each one being more detailed and more thorough than the previous ones.

[3] Do not look here for an acronym of any sort: our parser was simply named after the marsupial endemic to Australia, at considerable risk of extinction, and classified as *vulnerable*.

4.4.1 Correctness

Along section 4.4, we also aim to show that Algorithm 2 for LSCP is *correct by construction*. The notion of *correctness by construction* was first introduced by Dijkstra (1968), who,

> [a]s an alternative to methods by which the correctness of given programs can be established *a posteriori*, (. . .) proposes to control the process of program generation such as to produce *a priori* correct programs.

The underlying idea is to start from a problem specification and show that the solution is reached by successive application of mathematically correct rules. As opposed to usual formal verification, which proceeds *a posteriori* in order to prove that the algorithm conforms to the specification, the conclusion about correctness is obtained by showing that the algorithm is built correctly.

Applied to the problem we are concerned with, we come to the following description:

Correctness For any given input string of unrestricted natural language the LSCP finds, if it exits, the optimal spanning constituent licensed by the grammar.

By *licensed by the grammar* we mean that the solution parse is a loose model for the input string, and can be proven by predicate calculus in Ξ.

We adopt the principle suggested by Dijkstra of a constructive approach to designing an algorithm. In the upcoming sections, we give all the required elements in order to show how LSCP reaches a correct and optimal solution to our problem. The merit function in use will be described in due time (§4.4.4).

4.4.2 Sketch of the Process

We sketch here the parsing process as a configuration task.

The process starts off with an initial set of atomic constituents. The position of each word (*i.e.* an atomic constituent from the lexical construction) in the input string specifies how it may be connected to a preceding and/or a following constituent. The elementary step consists of first plugging together subsets of constituents into partial configurations. Each configuration is a k-subset[4] of the initial set[5]. Then the process of loose constraint satisfaction of the grammar is engaged, where each configuration is a possible assignment. finally, the legal configurations meeting our different assumptions plus conditions of optimality are projected into constituents by predicate calculus (see §3.2.5).

The elementary process is then iterated again with the new set of constituents, until a spanning constituent is reached. According to needs, the process can also include alternative, and eventually sub-optimal, solutions.

Procedure The parsing procedure is presented step by step below. Each step is then detailed in §4.4.3.

Step 0 (Initialisation) Every POS-tagged word in the input string is made an atomic constituent, and assigned maximum merit (1).

Step 1 (Selection) A reference span of consecutive surface words is decided, and all the existing constituents spanning part or all of it are selected.

The set of these constituents is $\mathcal{S}$. For the first iteration, the span contains only one word, and $\mathcal{S}$ contains one constituent for each possible POS-tag of this word. For a random iteration, the reference span contains $n > 1$ words, at most all the words in the input string; $\mathcal{S}$ contains every constituent spanning a subset of the span. None of the selected constituents in $\mathcal{S}$ may span words outside the reference span decided for this step. Note that this is not meant to aim for substructure relaxation: should a constituent be out of the span, then the configurations involved will be covered elsewhere, with a different span.

Step 2 (Enumeration) Every possible configuration of constituents in $\mathcal{S}$ over the reference span is enumerated.

Each constituent must have the exact same span as the reference one.

[4] According to Weisstein (2004): "A k-subset is a subset of a set on n elements containing exactly k elements".

[5] We will see later that there may be, in fact, more than one configuration for each k-subset.

Step 3 (Characterisation) Each configuration is characterised according to the grammar, and its merit is calculated.

The characterisation is a process of loose constraint satisfaction, where the grammar is a constraint system, and a configuration an assignment.

Step 4 (Projection) Each configuration is categorised, according to its characterisation, as an instance of one of the constructions specified in the grammar.

The configuration is said to be projected *into a construction. The result is a set of candidate constituents.*

Step 5 (Memoization) The best (optimal) constituent is memoized in the dynamic programming table π for the current span, while all the alternative candidates are relaxed from the search space.

(Iteration) Iterate the process again from Step 1, until the entire input string is spanned.

4.4.3 Algorithm

The algorithm for Loose Satisfaction Chart Parsing is based on the probabilistic Cocke-Kasami-Younger chart parsing algorithm (CKY), reported as Algorithm 1. The procedure described in §4.4.2 is implemented by Algorithm 2.

The skeleton of the CKY is augmented with a process of *loose constraint satisfaction*, which implements loose satisfaction as defined in §3.2.5. Loose constraint-based chart parsing also differs from the CKY in that no equivalent exists in PG for the Chomsky Normal Form of a grammar, which only generates binary trees. Subsequently, the baseline is generalised in order to cover n-ary structures. The third and last important difference is that the probabilities are replaced by constituent's costs, but we leave that aspect aside for the moment—§4.4.4 is dedicated to the merit function in use. The term *chart* is used here to refer to the use of a *dynamic programming table*.

Let us now detail Algorithm 2 a bit more.

Algorithm 1 Probabilistic CKY (Jurafsky and Martin, 2000, p. 455, corrected version)

```
function CKY($words, grammar$)
  returns the most probable parse and its probability

  Create and clear $\pi[num\_words, num\_words, num\_nonterminals]$

  /* base case */
  for $i \leftarrow 1$ to $num\_words$
      for $A \leftarrow 1$ to $num\_nonterminals$
          if $(A \rightarrow w_i)$ is in grammar then
             $\pi[i, i, A] \leftarrow P(A \rightarrow w_i)$

  /* recursive case */
  for $span \leftarrow 2$ to $num\_words$
      for $begin \leftarrow 1$ to $num\_words - span + 1$
          $end \leftarrow begin + span - 1$
          for $m \leftarrow begin$ to $end - 1$
              for $A \leftarrow 1$ to $num\_nonterminals$
              for $B \leftarrow 1$ to $num\_nonterminals$
              for $C \leftarrow 1$ to $num\_nonterminals$
                  $prob \leftarrow \pi[begin, m, B] \times \pi[m+1, end, C] \times P(A \rightarrow BC)$
                  if $(prob > \pi[begin, end, A])$ then
                     $\pi[begin, end, A] \leftarrow prob$
                     $back[begin, end, A] \leftarrow \{m, B, C\}$
  return build_tree($back[1, num\_words, S]$), $\pi[1, num\_words, S]$
```

The grammar must be in Chomsky Normal Form; the *back* array is used to store *backpointers* in order to enable the generation of the final parse tree(s); each backpointer value contains the index of the left-most daughter non-terminal, and the right-hand-side of the rule used.

Algorithm 2 Loose Satisfaction Chart Parsing

```
/* Initialisation */
Create and clear the chart $\pi$: every score in $\pi$ set to 0

/* Base case: populate $\pi$ with POS-tags for each word */
for $i \leftarrow 1$ to $num\_words$
    for (each POS-construction $T$ of $w_i$)
        if $\texttt{merit}(T) \geq \pi[i, 1, T]$ then
            Create constituent $w_i^T$, whose construction is $T$
            $\pi[i, 1, T] \leftarrow \{w_i^T, \texttt{merit}(w_i^T)\}$

/* Recursive case */
/* Step 1: SELECTION of the current reference span */
for $span \leftarrow 1$ to $num\_words$
    for $offset \leftarrow 1$ to $num\_words - span + 1$
        $end \leftarrow offset + span - 1$
        $K \leftarrow \emptyset$
/* Step 2: ENUMERATION of all the configurations */
        for (every set partition $\mathcal{P}$ in $[offset, \ldots, end]$)
            $K_{\mathcal{P}} \leftarrow \texttt{buildConfigurations}(\mathcal{P})$
            $K \leftarrow K \cup K_{\mathcal{P}}$
/* Step 3: CHARACTERISATION of the constraint system from the grammar */
            for (every configuration $\mathcal{A} \in K_{\mathcal{P}}$)
                $\chi_{\mathcal{A}} \leftarrow \texttt{characterisation}(\mathcal{A})$
/* Step 4: PROJECTION into constructions. */
                /* $\mathcal{C}_{\mathcal{A}}$ is a set of candidate constituents. */
                $\mathcal{C}_{\mathcal{A}} \leftarrow \texttt{projection}(\chi_{\mathcal{A}})$
                $\texttt{checkpoint}(\mathcal{C}_{\mathcal{A}})$
/* Step 5: MEMOIZATION of the optimal candidate constituent */
                for (every candidate constituent $x \in \mathcal{C}_{\mathcal{A}}$, of construction $C$)
                    if $\texttt{merit}(x) \geq \pi[offset, span, C]$ then
                        $\pi[offset, span, C] \leftarrow \{x, \texttt{merit}(x)\}$
        if $\pi[offset, span] = \emptyset$ then
            $\pi[offset, span] \leftarrow$ preferred forest in $K$
```

Dynamic Programming

LSCP relies, as the CKY and many other chart parsers, on *dynamic programming*. The part of the problem concerned with finding an optimal structure is divided into finding optimal solutions to sub-problems. These intermediate solutions are usually referred to as *optimal sub-structures*. Applied to our case, the fundamental principle met by dynamic programming is that *an optimal solution constituent structure is necessarily only made of sub-constituents, which all optimise the overall merit*. Assuming that the merit function has the right properties with respect to optimality (these properties are discussed §4.5) it is indeed quite straightforward to show, by contradiction, that if there exists a solution of optimal merit, and if one of its sub-constituents does not optimise this merit, then necessarily a different sub-constituent exists, which, if substituted to the sub-optimal one, yields a solution of better merit—this contradicts the hypothesis.

A *chart*, or *Dynamic Programming Table* (DP-Table), is used to store these optimal sub-structures. The storing process involved is called *memoization*. Dynamic programming also contributes to optimise the process in preventing multiple iterations of the same sub-problem to occur. The chart we are using (π) takes the words from the input string on the x-coordinate, and the span size on the y-coordinate. Table 4.1 illustrates how all the unlabelled sub-structures are covered using a DP-Table. Other examples of DP-Tables are illustrated in Tables 4.2 and 4.3.

Selection Step

The Selection Step iterates over the span size, from 1 to the number of words in the input string, and over the start position of the span (*offset*), so that every word in the string is covered by span. The iteration with $\{\mathit{offset}, \mathit{span}\} = \{i, j\}$ aims to solve the sub-problem which corresponds to filling π at the coordinates $\{i, j\}$. The selection $\mathcal{S}$ of constituents for the current span is made up of all the constituents contained in π at the coordinates $\pi[i, 1], \pi[i, 2], \ldots, \pi[i, j-1], \ldots, \pi[i+1, 1], \ldots, \pi[i+1, j-1], \ldots, \pi[\mathit{end}, j-1]$.

Note that the loop over *span* starts iterating from 1, thus allowing for constituents

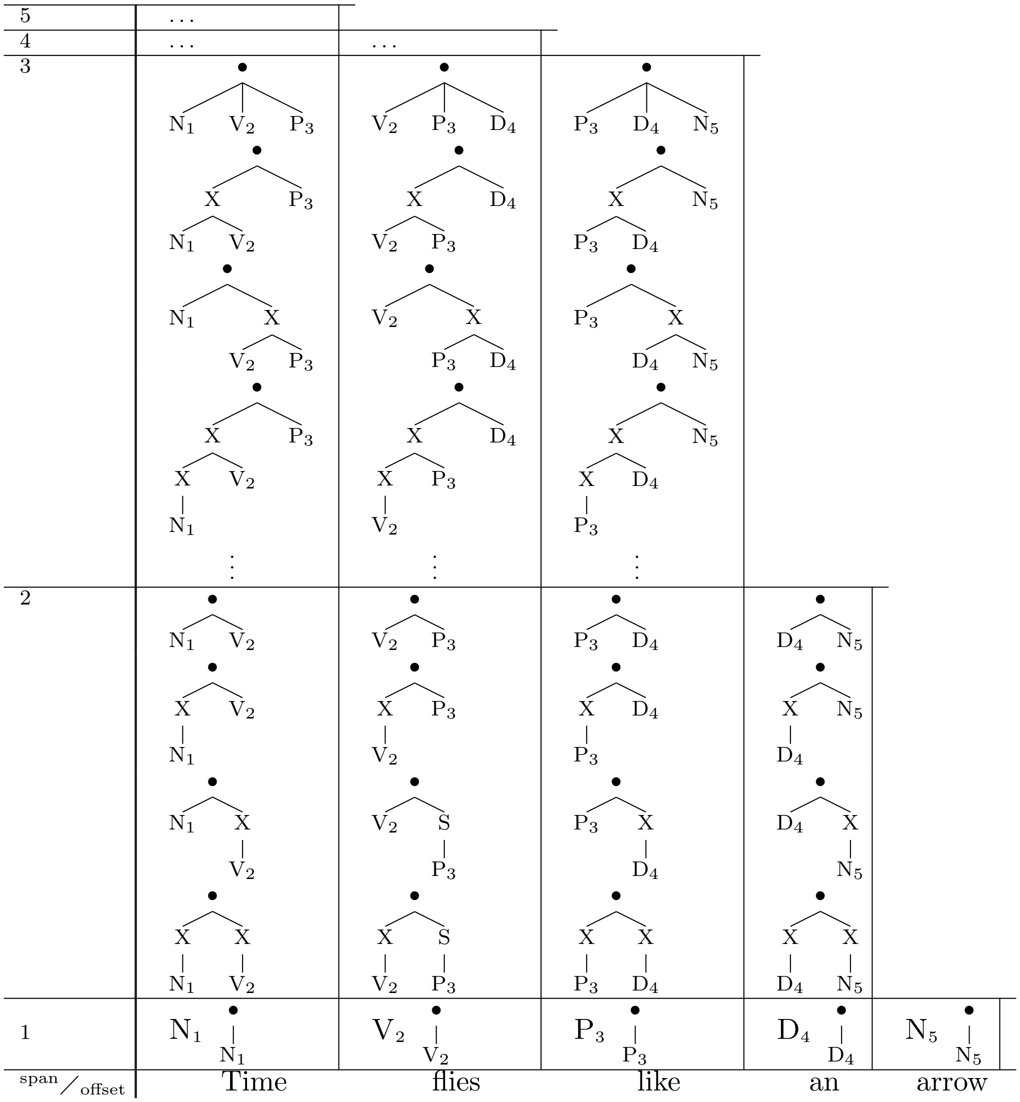


Table 4.1: Dynamic Programming Table, partially filled with unlabelled sub-structures, before being characterised and considered for memoization. *The symbol* • *represents an unlabelled root; X represents a labelled constituent—the labelling occurs during the* projection *Step. Note that the collection of sub-structures represented is* not *exhaustive.*

spanning one word only (*e.g.* an NP made of a single N), whereas the traditional CKY (Algorithm 1) starts iterating from 2. Starting from 1 implements the exception to Assumption 4, about a constituent arity.

Enumeration Step

We know that a valid configuration must span the words from *offset* to *end* (current span). If we consider a set partition[6] of $S = [\mathit{offset} \dots \mathit{end}]$, then each subset corresponds to a sub-problem already solved since its cardinality is necessarily less than the current span. Thus, the corresponding optimal sub-structure is already stored in π and can be retrieved. A configuration is obtained from a partition by combining together every sub-structure from the partition. Therefore, if we calculate all possible partitions of S we can make sure that we go through all possible configurations for the current span.

In fact, for the problem we are interested in, we do not need to calculate exactly *all* the set partitions. According to Assumption 5, we only need the partitions which are made up of subsets of contiguous elements.

Set Partitioning Algorithm 3 presents a solution to the problem of set partitioning. Given a finite set S of n elements, let us place a binary delimiter between every element, where a delimiter indicates a border between two subsets when set to 1, and no border when set to 0. Each combination of 0 and 1 for the delimiters corresponds to a partition. For n elements, there is $n-1$ possible positions for the delimiter. Therefore, all the partitions of S are enumerated by using the binary representation of all integers from 1 to 2^{n-2}. Note that the partition with all delimiters set to 0 is ignored, because meaningless in our case. In the end, the number of relevant partitions for S is $2^{n-2} - 1$.

Example 4.1 (Set Partitionning) *Let us consider the set* $s = \{a, b, c, d\}$ *of* n *words, spanning the words* w_i *to* w_j*, with* $j = i + n - 1$*. We want to enumerate*

[6] According to Weisstein (2006): "A *set partition* of a set S is a collection of disjoint subsets of S whose union is S".

Algorithm 3 Set Partitioning

```
function getSetPartitions($(\mathit{offset}, \mathit{span}, \pi)$)
          returns all the possible set partitions of $[offset, \ldots, \mathit{end}]$

  $\mathit{Partitions} \leftarrow \emptyset$
  $\mathit{end} \leftarrow \mathit{offset} + \mathit{span} - 1$
  for $\mathit{mask} \leftarrow 1$ to $(2^{span-1} - 1)$
      $\mathit{binMask} \leftarrow \mathit{mask}$ base 2
      $\mathcal{P} \leftarrow \emptyset$
      $i \leftarrow \mathit{offset}$
      /* for each digit $= 1_2$ in $\mathit{binMask}$ create a subset */
      for $j \leftarrow 1$ to $\mathit{span} - 1$
          if $(\mathit{binMask}[j] = 1_2)$ then
             $\mathcal{P} \leftarrow \mathcal{P} \cup \{\pi[i, j + \mathit{offset} - 1, X]\}$
             $i \leftarrow j + \mathit{offset}$
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{\pi[i, \mathit{end}, X]\}$
      $\mathit{Partitions} \leftarrow \mathit{Partitions} \cup \{\mathcal{P}\}$

  return $\mathit{Partitions}$
```

all the possible partitions of s. The list of all the partitions of s we are aiming for is the following:

$$\{\{a\}, \{b, c, d\}\}$$

$$\{\{a, b\}, \{c, d\}\}$$

$$\{\{a, b, c\}, \{d\}\}$$

$$\{\{a\}, \{b\}, \{c, d\}\}$$

$$\{\{a\}, \{b, c\}, \{d\}\}$$

$$\{\{a, b\}, \{c\}, \{d\}\}$$

$$\{\{a\}, \{b\}, \{c\}, \{d\}\}$$

Each partition is obtained by applying a binary mask of delimiters to s. We use the symbol $\oplus$ to represent the operation of applying a binary mask to a set. The solution partitions are thus obtained as follows:

$$\{a, b, c, d\} \oplus 001 \doteq \{\{a, b, c\}, \{d\}\}$$

$$\{a, b, c, d\} \oplus 010 \doteq \{\{a, b\}, \{c, d\}\}$$

$$\{a, b, c, d\} \oplus 101 \doteq \{\{a\}, \{b, c\}, \{d\}\}$$

$$\{a, b, c, d\} \oplus 111 \doteq \{\{a\}, \{b\}, \{c\}, \{d\}\}$$

$$\vdots$$

Enumerating the Labelled Configurations We have already seen how to retrieve the sub-structures from the chart π. What we have not said yet is that the Projection Step (Step 4) may categorise the same configuration into more than one construction, the optimal solution being memoized (Step 5) for every possible construction. The configuration must, therefore, be replicated as many times. This is the role[7] of `buildConfigurations` in Algorithm 2. In order to avoid confusions, when necessary an uncategorised (respectively *categorised*) configuration will be referred to as *unlabelled* (*labelled*) configuration. Algorithm 3 can easily be adapted to include these cases requiring replication.

Characterisation Step

The characterisation process is implemented with Algorithm 4, where an assignment is a configuration. More specifically, the assignment is made of the set of all the immediate constituents of the current unlabelled configuration. For the sake of the presentation, in Algorithm 2 the Projection Step is kept entirely separate from the characterisation, whereas it is in fact partly implemented as part of the function of characterisation, in order to save iterations.

Recall that in PG a *property* is a constraint. What the characterisation process needs to do is to check which constraints in the grammar ($\mathcal{G}$) are loosely satisfied by the assignment ($\mathcal{A}$). In order to ease the process, a lookup table is created for $\mathcal{G}$ when setting up the parser, which indexes every constraint by a *key*. This lookup table and the nature of this key are detailed in §4.4.5. For what we are interested in here, on the principle we must check, for every constraint in $\mathcal{G}$, whether $\mathcal{A}$ is an assignment for its variables. The algorithm actually proceeds the other way around; it calculates all possible combinations of variables from $\mathcal{A}$[8], then it uses each combination as a key in order to get a direct access to all constraints concerned. The combinations of variables we need are obtained by enumeration of all the binary representations of numbers in $[1..2^n - 1]$, where n is the cardinality of $\mathcal{A}$. Each binary number is then used as a mask over $\mathcal{A}$ in order to determine a combination. While iterating over the keys for $\mathcal{A}$, it is convenient to also retrieve from $\mathcal{G}$, for the same key, the constructions to be projected.

[7] `buildConfigurations` actually also combines the mechanism for keeping track of “backup” configurations, *i.e.* what could turn out to be part of a *forest of partial parses* in case everything else fails. These “backup” configurations come for handling what should otherwise be empty cells in the chart. Forests of partial parses are discussed later, in the Memoization Step section, p. 116.

[8] Rigorously, an assignment is not made of *variables* but of object instances to be assigned to variables, in our case *constituents*.

Algorithm 4 Characterisation Function

```
function characterisation($\mathcal{A} = \langle c_1, \ldots, c_n \rangle$ : assignment, $\mathcal{G}$: grammar)
            returns the set of evaluated properties relevant to $\mathcal{A}$,
                    and the set of projected constructions for $\mathcal{A}$.

  /* For storing the result characterisation: */
  create and clear $\chi_{\mathcal{A}}[property]$: table of boolean, indexed by property

  /* For storing the result projected constructions: */
  create and clear $\mathcal{C}_{\mathcal{A}}$: set of construction

  /* For temporarily storing the properties to be evaluated: */
  create and clear $S$: set of property

  for ($mask \in [1 \ldots 2^n - 1]$)
      $key \leftarrow$ applyBinaryMask($\mathcal{A}$, $mask$)
      if ($key$ is in the set of indexes for $\mathcal{G}$)  then
         /* Properties are retrieved from the grammar, then evaluated */
         $S \leftarrow \mathcal{G}[key]$.getProperties()
         $\chi_{\mathcal{A}} \leftarrow$ evaluate($S$)

         /* Projection Step: retrieval of the constructions to be projected */
         $\mathcal{C}_{\mathcal{A}} \leftarrow \mathcal{G}[key]$.getDominantConstructions()

  return $\chi_{\mathcal{A}}, \mathcal{C}_{\mathcal{A}}$
```

The *key* determined by `applyBinaryMask` is a combination of constructions (*i.e.* those constructions in $\mathcal{A}$ for which the corresponding bit in the mask is set to 1); it is used, after application of a hash function, as an index for retrieving, from a lookup table, the constraints in the grammar this combination is concerned with.

Although it looks like constraints are reevaluated every time that the same constraint type must be evaluated for the same constructions, these reevaluations are only redundand in the case of monotonic constraints (*i.e.* Linearity and Dependency). For all the others, since the context of a different configuration means a different assignment every time, these constraints may evaluate differently under each of them, hence the need for reevaluation.

Projection Step

Conceptually, the Projection Step consists of making a *judgement* about the Construction of a constituent (see §3.2.4). It is seen as a categorisation problem, where an unlabelled configuration is categorised into a construction, according to its characterisation. More practically, it is a matter of labelling unlabelled configurations.

In §3.2.4 we have presented a grammar as a collection of constructions, where each construction is specified by a set of constraints. The operation we need to perform now requires to reverse the information, in order to determine which constructions are projected (*i.e.* inferred) given the knowledge we have of the set of constraints which are loosely satisfied by a configuration–*i.e.*, its characterisation. A lookup table is created during the setup phase of *Numbat*, which let us retrieve these constructions directly, a constraint serving as a key. This part of the process is the one actually implemented as part of the function of characterisation (§4.4.3).

The other part of the process, implemented in the `projection`() function from Algorithm 2, is concerned with enforcing the different assumptions we stated in §4.3.2. The sub-structures not meeting these assumptions are systematically relaxed.

Memoization Step

The Memoization Step aims to store the optimal constituent for each possible construction for a given cell in the chart. The merit function in use for optimality is presented and discussed §4.4.4.

Should the current cell in the chart not being populated with any constituents, a preferred *forest of partial parses* is used instead, and populates the cell. The preferred forest is identified on the fly, while going through all the possible configurations (as part of `buildConfigurations` in the Enumeration Step); a pointer is maintained to the preferred configuration during enumeration. The preference goes to:

- constituents with the widest span;
- the least overall number of constituents.

This translates heuristically into a *preference score* computed as follows (where F is the forest, C_i the constituents it is made up of, $\texttt{merit}(C_i)$ the merit of a constituent, and p_F the preference score associated with F):

$$p_F = \mathit{span} \cdot (\texttt{merit}(C_i) + \mathit{span})$$

p_F should be seen as "the score of the last chance": when the main parsing process fails to find a dominant construction for a set of constituents the different configurations of these constituents somehow compete to be chosen as the default partial parse (for the span at stake). The winner is the one with the best preference score. Note that in the worst-case scenario a partial parse has the same granularity as the parser's input—but not necessarily; as far as *Numbat* is concerned we are left with a sequence of Part-Of-Speech constituents. Thus the LSCP algorithm, and *Numbat* more specifically for that matter, always delivers a parse for any input.

Quite obviously, as a heuristic p_F is only one of the many possiblities for computing such a preference, and it would be interesting to investigate in details some of these possibilities and see which one gives the best results. However, what exactly makes a forest of partial parses better than another one is unclear. For the purpose of experimenting with gradience, since the corpus we are working with was artificially constructed with sentences built according to specific error patterns, only complete parses are predicted to be output by the parser; thus forests of partial parses are seen (and evaluated) as errors (see Chapter 5 for experiments on gradience and §4.9 for *Numbat*'s evaluation). While focusing on what the default output should look like in case of an error was not our prime concern, there might be applications where it is. Further investigation should look at different avenues according to needs.

One option is to look at different conjectures than widest span and the least possible constituents, and/or different ways to capture them numerically. Maybe the merit of each constituent should not be taken into account, for instance. Or on the contrary, priority could be given to well-formed constituents.

Another option is to start introducing weights; first to constructions, then—more ambitiously—to constraints. If the task of weighting constraints is known for being labour-intensive (see, for instance, Keller, 2000 and Schröder et al., 2000, whose works we have reviewed in Chapter 2), weighting constructions, on the other hand, should not require as much work. First, because their number is by far less important than the number of constraints, and second because it could probably be taken advantage of corpora, where constituents are annotated with frequencies. It is likely that these frequencies could be used to provide a good default shallow parse.

Note that more information could have probably been memoized, like the set partitions, or the monotonic constraints. The memoization of monotonic constraints, in particular, could probably be inspired from part of VanRullen's SeedParser (in compiling them, for example).

4.4.4 Merit Function

We have not discussed so far the merit function in use in *Numbat*. Its purpose may be interpreted in two different, yet complementary ways. One way is to serve as an *optimality function*, in order to identify, preferably uniquely, which characterised constituent structure best describes the syntactic characteristics of an input string. The main question the function must answer, then, concerns the criteria such a decision of optimality is based upon.

Another interpretation comes from that the score is absolute and assigned to a constituent, that is, with respect to a specific construction. Therefore a given input string may be analysed in as many different constituents, and may be assigned as many different scores as there are constructions it can be categorisied into. These scores may then be compared with each other and used to discriminate which class (*i.e.* construction) the input item belongs to. In doing so, that merit function is seen as a means to address the categorisation problem referred to as *Intersective Gradience* (IG) by Aarts (see §2.2.2). As a reminder, IG is concerned with uniquely categorising an item, which presents characteristics from more than one class. Even though Aarts focuses on standard (*i.e.* non-deviant) language, conceptually the problem can be generalised to tackle cases of ill-formedness, where an utterance may also be categorised in more than one construction. In such cases, a decision process is required in order to unambiguously categorise the utterance. A word of caution ought to come with the *unambiguous* aspect of the categorisation though. Indeed, it is well-known that even standard language may be inherently syntactically ambiguous. Although the merit function might also be, to some extent, well-suited to address this problem, it is not our purpose here and we, thus, leave that question open. The main question the merit function must answer, again, concerns the criteria involved in this decision-making process.

These two interpretations are in fact complementary, in that unambiguous categorisation is achieved through optimality. In other words, a solution parse is considered as the best one because it provides an optimal categorisation of an item compared to the alternative possible categorisations.

Whether in Aarts' address of IG, or in OT (Prince and Smolensky, 1993) or LOT (Keller, 2000), constraint violations only are considered meaningful for addressing that problem. We take a different standing point, since we believe that more than simply accounting for absent or broken characteristics, the problem we are faced with concerns finding out the most *cohesive* syntactic description of an utterance. By *cohesive* we mean that a syntactic description should be assessed not only for the rules or constraints it breaks, but also for those it meets. In that sense, the greater the proportion of properties it meets, the more cohesive the syntactic representation. When all properties are met the cohesion equals its absolute maximum and the utterance is syntactically well-formed. And conversely, an ill-formed utterance does not meet all properties and the cohesion of its syntactic representation is strictly less than the absolute maximum.

In contrast, in OT-like approaches the set of all the contsraints in the grammar is typically inconsistent, which makes it impossible for all constraints to be satisfied. Therefore, the structures defined as grammatical are not those satisfying all constraints. This is a major impediment to discriminating between deviant and non-deviant language. As a consequence, gradience is modelled in these approaches as being ubiquituous among Optimality-Theoretically grammatical structures, which differs from our own conception of gradience.

Counting Constraints Relying on constraint counts, whether for optimality purpose or for modelling gradience, raises questions regarding what is accounted for. One question is concerned with the granularity of the material being counted, and another one is concerned with how the cardinality of constraint sets is figured. As for granularity, one may wonder whether a grammar statement such as "in French, when the Direct Object is placed before the auxiliary verb *avoir* (*to have*) the past participle must agree in number and in gender with the Subject" should be counted as one, maybe two, or even three or up to four different statements. Indeed, it could be seen as a single statement, but it could be split in two: "the Direct Object is placed before the auxiliary verb *avoir*", and[9] "the past participle must agree in gender and in person with the Subject", or it could also be splitted even further with respect to the agreement, and so on. We took the party of formulating constraints with the finer possible granularity allowed by the formalism, while maintaining consistency in the grammar—especially at the construction level. To follow up with the same example, the Direct Object being placed before the auxiliary verb *avoir* would count as one, the agreement as two, and the implication as one, the initial statement thus amounting to four different constraints[10]. Our grammars were written so that every specified constraint is as atomic as possible with PG, in the sense that no variable changes would split it in two constraints or more. This was done in order to avoid non-linearities that may be caused by changes of variables and may distort the constraint counts.

[9] Of course, "and" should not be taken here as a conjunction from a strictly logical point-of-view, since an implication of the form $A \Rightarrow B$ is equivalent to the disjunction $\neg A \vee B$; it is rather a matter of splitting one statement into two different ones.

[10] In practice, however, we did not implement the constraint involving the Direct Object; the reason for that being that we did not implement any functions like Direct Object.

As for cardinality, the question is to decide whether a count of evaluated constraints should account for all instances or, alternatively, for one instance of each constraint. we opted for counting every instance. In Chapter 3 we have illustrated the role of quantifier-free variables with an example[11] where the fragment "The day the" is analysed as part of the sentence "The day the student graduates". In that example, two instances of the same constraint D ≺ N are possible:

$$\text{The}_D \prec \text{day}_N \tag{4.1}$$

$$\text{day}_N \prec \text{the}_D \tag{4.2}$$

A rationale for the alternative option would be to argue that, for example, at the sentence level multiple violations of the same constraint may not necessarily induce a decrease of gradience strictly proportional to the number of violations. For example, in the utterances 4.3 to 4.5 below,

(4.3) *She prepare the cake

(4.4) *She bake and eat the cake

(4.5) *She prepare, bake, and eat the cake

it is not obvious whether the degree of syntactic gradience decreases proportionally to the number of verbs violating the person agreement. While the argument whereby such recurrent error patterns are associated with some form of attenuation effect might be relevant to discussions on gradience and the categorisation problem mentioned earlier in this section, it is not involved in the optimality problem which we discussed alongside and which is concerned with finding an optimal parse for each of these utterances. It is the latter that we address with the merit function being discussed in this section.

[11] Sentence 3.54, p. 99.

Definitions

As mentioned earlier, this merit aims to capture the cohesion of a constituent; but it is also expected to meet the requirement we have about the optimality of the solution constituent, and to allow the overall merit to be optimised by partial solutions (substructures). The function we are using was first defined in VanRullen (2005), and referred to as *density of satisfaction*[12]. Although we define it here from a slightly different perspective in order to emphasise the proportion we have been discussing, the calculation is exactly the same.

Definition 4.2 (Cohesion, part 1) *Let V be a constituent, $\chi(x_1, \ldots, x_n) = \chi^+(x_1, \ldots, x_n) \cup \chi^-(x_1, \ldots, x_n)$ its characterisation, where $x_1, \ldots, x_n$ are the constituents that V immediately dominates. We note:*

- *P the number of constraints in χ, and P^+ and P^- the numbers of satisfied and violated constraints respectively, with $P = P^- + P^+$*
- *$n = \mathrm{card}(x_1, \ldots, x_n)$ the number of constituents V dominates.*

Note that P only accounts for the constraints applying to the immediate constituents of V. Each of the x_i constituents being itself characterised unless it is from an atomic construction (POS), a recursive function is now defined in order to sum up all constraints V is concerned with.

Definition 4.3 (Cohesion, part 2) *The total number $\mathcal{P}_V^+$ (respectively $\mathcal{P}_V^-$) of constraints the constituent V satisfies (violates) is defined as follows:*

$$\mathcal{P}_V^+ = \begin{cases} 0 & \textit{if POS} \\ P^+ + \sum_{i=1}^n (\mathcal{P}_{x_i}^+) & \textit{otherwise} \end{cases}$$

$$\mathcal{P}_V^- = \begin{cases} 0 & \textit{if POS} \\ P^- + \sum_{i=1}^n (\mathcal{P}_{x_i}^-) & \textit{otherwise} \end{cases}$$

[12] The term *density* is used ambiguously in the literature about PG (Blache and Prost, 2005; Blache, Hemforth, and Rauzy, 2006; VanRullen, 2005), because associated with different numerical definitions. In order to avoid confusions, we simply avoid the term, unless for explicit references.

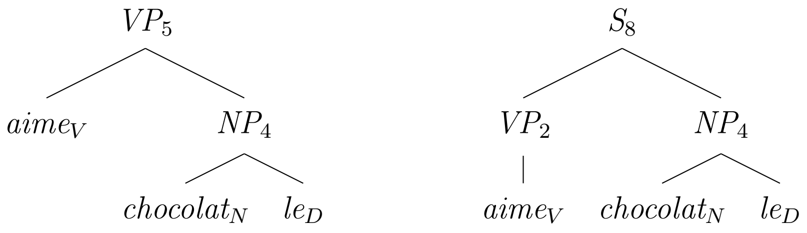
Now, we can define the *cohesion* of V as the proportion of satisfied constraints[13].

Definition 4.4 (Cohesion) *The* cohesion $\mathcal{C}_V$ *of the constituent* V *is defined by the following ratio:*

$$\mathcal{C}_V = \frac{\mathcal{P}_V^+}{\mathcal{P}_V^- + \mathcal{P}_V^+}$$

A score of cohesion takes its value in $[0 \dots 1]$.

Example 4.2 (Cohesion score) *In Example 4.4, the chart* π *contains the two candidate constituents* VP_5 *and* S_8 *at the coordinates* $\pi[2][3]$ *represented below:*



These two constituents are characterised as follows ($\mathcal{A}_5$ and $\mathcal{A}_7$ respectively):

$$\begin{array}{rcl}
\chi^+_{\mathrm{VP}_5} & = & \chi^+_{\mathcal{A}_5} & = & \{\textit{(P3.19)}, \textit{(P3.21)}, \textit{(P3.23)}\} \\
\chi^+_{\mathrm{S}_8} & = & \chi^+_{\mathcal{A}_7} & = & \{\textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.5)}\} \\
\chi^-_{\mathrm{S}_8} & = & \chi^-_{\mathcal{A}_7} & = & \{\textit{(P3.4)}\}
\end{array}$$

VP_5 *and* S_8 *contain the constituents* VP_2 *and* NP_4 *characterised themselves as follows (with $\mathcal{A}_2$ and $\mathcal{A}_4$ respectively):*

$$\begin{array}{rcl}
\chi^+_{\mathrm{VP}_2} & = & \chi^+_{\mathcal{A}_2} & = & \{\textit{(P3.19)}\} \\
\chi^+_{\mathrm{NP}_4} & = & \chi^+_{\mathcal{A}_4} & = & \{\textit{(P3.6)}, \textit{(P3.8)}, \textit{(P3.7)}, \textit{(P3.15)}, \textit{(P3.17)}, \textit{(P3.18)}\} \\
\chi^-_{\mathrm{NP}_4} & = & \chi^-_{\mathcal{A}_4} & = & \{\textit{(P3.11)}\}
\end{array}$$

[13] It was pointed out to us by Gerald Penn (examiner, *rapporteur*) that out of completeness a dual should be defined as well, in which $\mathcal{P}_V^-$ is weighted against all of the satisfied constraints. While this is true, the dual cohesion is of no use to us.

The cohesion scores are then the following:

$$\begin{aligned}
\mathcal{C}_{\mathrm{NP}_4} &= \frac{6}{6+1} &= 0.86\\
\mathcal{C}_{\mathrm{VP}_2} & &= 1\\
\mathcal{C}_{\mathrm{S}_8} &= \frac{4+1+6}{11+1+1} &= 0.79\\
\mathcal{C}_{\mathrm{VP}_5} &= \frac{3+6}{9+1} &= 0.9
\end{aligned}$$

4.4.5 Consulting The Grammar

Because the grammar is used for different purposes, in different ways, and at different steps in the parsing process, different indexing tables are created during the system setup, in order to ease looking up information in the grammar. These tables save processing effort, and thus reduce the complexity of the LSCP. We define below different lookup tables for the grammar.

Constraint Index An assignment rarely—if ever—concerns all variables in the constraint system. Therefore, we index the constraints by a hashed key on the variables they contain, in order to access them directly. More precisely, the key contains the free variables and the variables quantified universally. Indeed, the variables quantified existentially may be absent from the assignment. In case of Requirement or Exclusion, for instance, the point is precisely to check whether or not the existential variables are present in the assignment.

As an example, the constraint D $\prec$ N is indexed by the pair $\{D, N\}$ (more exactly a hash of it), and the constraint N $\Rightarrow$ D is indexed by the singleton $\{N\}$.

Projection Index We have seen in §3.2.4 that the constructions in the grammar can be projected (*i.e.* inferred) by any of the constraints they are specified by. Initially, the grammar is a set of constructions, each being specified by a set of constraints; in order to look up which construction(s) a given constraint specifies, each construction is indexed by as many constraints as those contained in its specification. In other words, each constraint in the grammar indexes all the constructions it specifies. This way, the projection Step in the parsing process, concerned with inferring dominant constituents, is only a matter of direct access to the lookup table.

Figure 4.1 exemplifies how the two indexes (constraint and projection) are implemented into one lookup table.

Index	Constraints	Projected Constructions
$(\mathrm{N}, \mathrm{PP})$	$\mathrm{N} \prec \mathrm{PP}$	NP
$(\mathrm{V}, \mathrm{PP})$	$\mathrm{V} \prec \mathrm{PP}$	VP
$(\textsc{Adv}, \mathrm{A})$	$\textsc{Adv} \prec \mathrm{A}$	Sup, AP
$(\mathrm{V}, \mathrm{NP})$	$\mathrm{V} \not\Leftrightarrow \mathrm{NP}$ $\mathrm{V} \prec \mathrm{NP}$	VP, Cleft
(VP)	$\mathrm{VP}!$ $\triangle\mathrm{VP}$	Rel, Compl, S, Cleft

Figure 4.1: Example of lookup table for the grammar (sample)

4.4.6 Algorithm Walkthrough

In order to illustrate the parsing process, we are now going to walk through the algorithm. The process is also exemplified in Example 4.3 (along with Table 4.2) for a well-formed utterance, and in Example 4.4 (along with Table 4.3) for a deviant utterance.

Example 4.3 (Parsing *Chloé* . . . (well-formed): characterisations) *The characterisations for the DP-Table presented in Table 4.2 are given below.*

$$\begin{array}{rcrcl}
\pi[1][1] & : & \mathcal{A}_1 & = & \langle Chloe_N \rangle \\
 & & \chi^+_{\mathcal{A}_1} & = & \big\{ \textit{(P3.6)}, \textit{(P3.8)}, \textit{(P3.17)} \big\} \\
\pi[2][1] & : & \mathcal{A}_2 & = & \langle aime_V \rangle \\
 & & \chi^+_{\mathcal{A}_2} & = & \big\{ \textit{(P3.19)} \big\} \\
\pi[1][2] & : & \mathcal{A}_3 & = & \langle Chloe_N, aime_V \rangle \\
 & & \chi^+_{\mathcal{A}_3} & = & \big\{ \textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.4)}, \textit{(P3.5)} \big\} \\
\pi[3][2] & : & \mathcal{A}_4 & = & \langle le_D, chocolat_N \rangle \\
 & & \chi^+_{\mathcal{A}_4} & = & \big\{ \textit{(P3.6)}, \textit{(P3.8)}, \textit{(P3.7)}, \textit{(P3.11)}, \textit{(P3.15)}, \textit{(P3.17)}, \textit{(P3.18)} \big\} \\
\pi[2][3] & : & \mathcal{A}_5 & = & \langle aime_V, NP_4 \rangle \\
 & & \chi^+_{\mathcal{A}_5} & = & \big\{ \textit{(P3.19)}, \textit{(P3.21)}, \textit{(P3.23)} \big\} \\
\pi[1][4] & : & \mathcal{A}_5 & = & \langle NP_1, VP_5 \rangle \\
 & & \chi^+_{\mathcal{A}_6} & = & \big\{ \textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.4)}, \textit{(P3.5)} \big\}
\end{array}$$

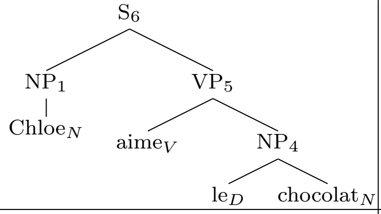
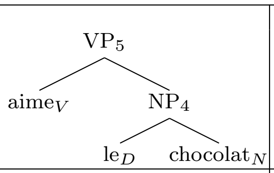
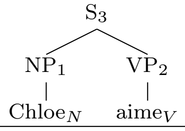
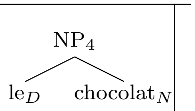
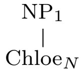
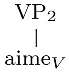
4	 S_6 NP_1 VP_5 $Chloe_N$ $aime_V$ NP_4 le_D $chocolat_N$			
3	nil			
2		nil		
1	 $Chloe_N$	 $aime_V$	le_D	$chocolat_N$
span/offset	Chloé	aime	le	chocolat

Table 4.2: Parsing *Chloé aime le chocolat (* Chloe likes [the] chocolate*)*: the final DP-Table, populated with constituents

Example 4.4 (Parsing *Chloé* . . . (deviant): characterisations) *The characterisations for the chart presented in Table 4.3 are given below.*

$$\begin{array}{rcrcl}
\pi[1][1] & : & \mathcal{A}_1 & = & \langle Chloe_N \rangle \\
 & & \chi^+_{\mathcal{A}_1} & = & \big\{ \textit{(P3.6)}, \textit{(P3.8)}, \textit{(P3.17)} \big\} \\
\pi[2][1] & : & \mathcal{A}_2 & = & \langle aime_V \rangle \\
 & & \chi^+_{\mathcal{A}_2} & = & \big\{ \textit{(P3.19)} \big\} \\
\pi[1][2] & : & \mathcal{A}_3 & = & \langle Chloe_N, aime_V \rangle \\
 & & \chi^+_{\mathcal{A}_3} & = & \big\{ \textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.4)}, \textit{(P3.5)} \big\} \\
\pi[3][2] & : & \mathcal{A}_4 & = & \langle chocolat_N, le_D \rangle \\
 & & \chi^+_{\mathcal{A}_4} & = & \big\{ \textit{(P3.6)}, \textit{(P3.8)}, \textit{(P3.7)}, \textit{(P3.15)}, \textit{(P3.17)}, \textit{(P3.18)} \big\} \\
 & & \chi^-_{\mathcal{A}_4} & = & \big\{ \textit{(P3.11)} \big\} \\
\pi[2][3] & : & \mathcal{A}_5 & = & \langle aime_V, \mathrm{NP}_4 \rangle \\
 & & \chi^+_{\mathcal{A}_5} & = & \big\{ \textit{(P3.19)}, \textit{(P3.21)}, \textit{(P3.23)} \big\} \\
 & : & \mathcal{A}_7 & = & \langle \mathrm{VP}_2, \mathrm{NP}_4 \rangle \\
 & & \chi^+_{\mathcal{A}_7} & = & \big\{ \textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.5)} \big\} \\
 & & \chi^-_{\mathcal{A}_7} & = & \big\{ \textit{(P3.4)} \big\} \\
\pi[1][4] & : & \mathcal{A}_5 & = & \langle \mathrm{NP}_1, \mathrm{VP}_5 \rangle \\
 & & \chi^+_{\mathcal{A}_6} & = & \big\{ \textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.4)}, \textit{(P3.5)} \big\}
\end{array}$$

We consider that every word in the input string is annotated with one and one POS-tag only. Note that generalising the process in order to handle multiple alternative POS for each input word is rather straightforward, since POS-tags are dealt with as lexical constructions.

For the sake of the presentation, the selection of constituents is determined by a *window* placed over the input string, which delimits the current span. The window acts as a hypothetical model (assignment) to be checked against the grammar. In order to make sure that no sub-structure is ignored the algorithm iterates over the number of words the window spans, from 1 to the size n of the input string. All possibilities are considered by going through all the 5 steps for every position and every size of

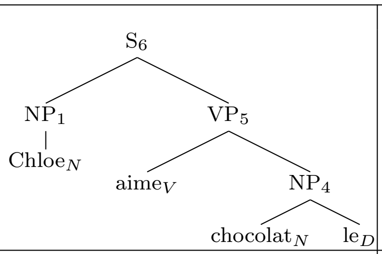
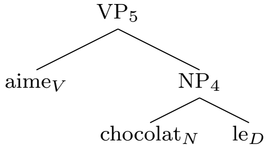
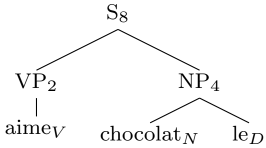
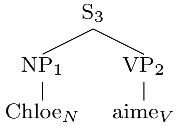
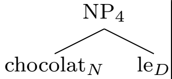
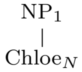
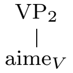
4				
3	nil	 		
2		nil		
1	 Chloe_N	 aime_V	chocolat_N	le_D
span / offset	Chloé	aime	chocolat	le

Table 4.3: Parsing **Chloé aime chocolat le (*Chloe likes chocolate [the]*)*: the final DP-table, populated with constituents

the window[14]. We proceed bottom-up, from left to right—although nothing prevents proceeding from right to left.

Going through Steps 1 to 5 of the algorithm with a window positioned at the offset *begin* and of size *span* corresponds to populating the DP-Table π at the coordinates $[offset][span]$. To illustrate the process as it goes, we use $span = 3$.

Step 1 (Selection) The window is positioned over the words w_i to w_{i+2}. Therefore, at the end of Step 5 we will have populated $\pi[i][2]$.

Step 2 (Enumeration) For $span_{i,i+2}$ Algorithm 3 enumerates 3 partitions, corresponding to 3 assignments to collect from π:

$$\{\{a\},\{b\},\{c\}\} = \{\pi[i][1], \pi[i+1][1], \pi[i+2][1]\} \tag{4.6}$$

$$\{\{a\},\{b,c\}\} = \{\pi[i][1], \pi[i+1][2]\} \tag{4.7}$$

$$\{\{a,b\},\{c\}\} = \{\pi[i][2], \pi[i+2][1]\} \tag{4.8}$$

For $span = 3$ we have the following 3 assignments:

$$\mathcal{A} = \{C_{i,1}, C_{i+1,1}, C_{i+2,1}\} \tag{4.9}$$

$$\mathcal{A}' = \{C_{i,1}, C_{i+1,2}\} \tag{4.10}$$

$$\mathcal{A}'' = \{C_{i,2}, C_{i+1,1}\} \tag{4.11}$$

Example 4.5 (Access to π) *Let us consider the partition $p = \{\{a\},\{b,c\},\{d\}\}$, and an offset of 3. If we note π the DP table, the elements in π are indexed by a pair of coordinates $[x][y]$ where x is the absolute position of the subset (i.e. offset + relative position in the partition), and y the span (i.e. the cardinality of the subset). Thus, in the case of p, the first element $\{a\}$ corresponds to the sub-trees stored at $\pi[3][1]$; the second element $\{b,c\}$ corresponds to $\pi[4][2]$; and the third element $\{d\}$ corresponds to $\pi[6][1]$.*

[14] Echoing the discussion in §3.3.4 it is important to emphasise that Algorithm 2 for LSCP is *not* incremental. As we will see later in §4.4.6, the process of constraint satisfaction we are implementing only considers assignments of fixed size, which are not incremented dynamically.

Step 3 (Characterisation) Each assignment corresponds to a specific configuration of constituents. For each assignment $\mathcal{A}, \mathcal{A}', \mathcal{A}''$ we look up in the grammar (see §4.4.5) which constraints are evaluable, and we instantiate them according to the assignment. The evaluation of each instantiated constraint can then take place. If we note $\phi_{\mathcal{A}}, \phi_{\mathcal{A}'}, \phi_{\mathcal{A}''}$ respectively the constraints after instantiation, we have:

$$\mathcal{A} \models\!\!\!\!\simeq \phi_{\mathcal{A}}, \mathcal{A}' \models\!\!\!\!\simeq \phi_{\mathcal{A}'}, \mathcal{A}'' \models\!\!\!\!\simeq \phi_{\mathcal{A}''}$$

As a result, each configuration is now characterised (see Definition 3.14) as follows (where $\chi_{\mathcal{A}}, \chi_{\mathcal{A}'}, \chi_{\mathcal{A}''}$ are the characterisations of $\mathcal{A}, \mathcal{A}', \mathcal{A}''$ respectively):

$$\begin{aligned} \chi_{\mathcal{A}} &= \langle \chi^{+}_{\mathcal{A}}, \chi^{-}_{\mathcal{A}} \rangle & (4.12) \\ \chi_{\mathcal{A}'} &= \langle \chi^{+}_{\mathcal{A}'}, \chi^{-}_{\mathcal{A}'} \rangle & (4.13) \\ \chi_{\mathcal{A}''} &= \langle \chi^{+}_{\mathcal{A}''}, \chi^{-}_{\mathcal{A}''} \rangle & (4.14) \end{aligned}$$

Example 4.6 (Characterisation) *In reference to Example 4.4, consider the span* aime chocolat le *(word-to-word* likes chocolate the*), and the assignment* $\mathcal{A}_7 = \langle \mathrm{VP}_2, \mathrm{NP}_4 \rangle$ *where* VP_2 *is the constituent* $(\mathrm{aime})_{VP}$ *and* NP_4 *the constituent* $(\mathrm{chocolat\ le})_{NP}$*; the toy grammar* Γ *from §3.2.4 tells us that the following constraints are evaluable:*

$$\phi_{\mathcal{A}_7} = \{\textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.5)}, \textit{(P3.4)}\}$$

Out of these, the following ones are satisfied:

$$\chi^{+}_{\mathcal{A}_7} = \{\textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.5)}\}$$

while the following one is violated:

$$\chi^{-}_{\mathcal{A}_7} = \{\textit{(P3.4)}\}$$

Step 4 (Projection) For each atomic constraint in $\phi_{\mathcal{A}}$ (respectively in $\phi_{\mathcal{A}'}$ and $\phi_{\mathcal{A}''}$) we look up in the grammar which construction(s) they specify. Each pair

$\langle$constraint, Construction$\rangle$ correponds to a projection rule (see §3.2.4). Thus to every instantiated constraint $\phi_{\mathcal{A}}, \phi_{\mathcal{A}'}, \phi_{\mathcal{A}''}$ corresponds a set of constructions which can be projected for this assignment. A new constituent is constructed for every distinct pair $\langle$Assignment, Construction$\rangle$.

Example 4.7 (Projection (single)) *Following up on Example 4.6, according to* Γ *the following rules apply to the constraints in* $\phi_{\mathcal{A}_7}$*:*

$$\text{AP} \longrightarrow \bigwedge\{\textit{(P3.1)}, \textit{(P3.2)}, \textit{(P3.3)}, \textit{(P3.4)}, \textit{(P3.5)}\}$$

Example 4.8 (Projection (multiple)) *In order to illustrate a case where more than one construction can be projected, consider now the hypothetical assignment* $\mathcal{B} = \langle \text{D}_1, \text{A}_2 \rangle$*, and* Γ*;* $\mathcal{B}$ *is an assignment for the following constraint:*

$$\phi_{\mathcal{B}} = \{\textit{(P3.7)}, \textit{(P3.30)}, \textit{(P3.31)}, \textit{(P3.36)}\}$$

The following rules apply to subsets of $\mathcal{B}$*:*

$$NP \longrightarrow \textit{(P3.7)} \tag{4.15}$$

$$AP \longrightarrow \bigwedge\{\textit{(P3.30)}, \textit{(P3.31)}, \textit{(P3.36)}\} \tag{4.16}$$

Step 5 (Memoization) Every constituent C paired with the assignment $\mathcal{A}$ (*idem* for $\mathcal{A}'$ and $\mathcal{A}''$) is memoized in the chart (with $\textsc{Cat}_1$ the construction of C):

```
if $\texttt{merit}(\chi_{\mathcal{A}}) \geq \pi[\textit{begin}][\textit{span}][\textsc{Cat}_1]$ then
  $\pi[\textit{begin}][\textit{span}][\textsc{Cat}_1] \leftarrow C$
```

The constituents for which the memoization fails are relaxed from the search space.

(Iteration) Steps 1 to 4 are iterated for each offset of $\text{span}_{i,j}$. When the window reaches the end of the string, the offset is set to 1, and the size of $\text{span}(i, j)$ is incremented.

4.5 Optimality

What we aime to show in this section is that the Cohesion function presents all the suitable and required algebraic properties to ensure the optimality of the solution parse generated by our algorithm for LSCP. We already know that the algorithm goes through all the smaller constituents (*i.e.* with the smaller span) first before it moves on and generates the larger ones. Since the algorithm only considers immediate sub-constituents when building a new constituent, what we need to show with respect to the merit function in use—*i.e.* Cohesion—is that the influence of grand-children exclusively factors through the influence of the immediate descendants of the new constituent. If it is not the case, that is to say, if the Cohesion score is influenced by grand-children's scores, then for optimality to be enforced it would require the algorithm to test the score of a newly created constituent against those of its grand-children as well. Such a factorisation restriction, along with the Memoization Step which only memoizes a constituent **if** `merit`$(C) \geq \pi[\mathit{offset}, \mathit{span}, C]$, establishes by force that the solution generated by our algorithm is optimal.

As a preamble, notice that during memoization the Cohesion score is only used to compare constituents of both *same span* and *same construction*. If two constituents differ on one or the other, than their cohesion can not be compared with respect to optimality; both of them are stored in the chart. This means that we are only looking for disambiguation, and not for speed.

The Cohesion of a constituent is formulated as the ratio of constraints which hold true for this constituent, by the total number of constraints which either hold or fail for the constituent (see §4.4.4 for more details). The Cohesion score of a constituent thus does not directly depend on the Cohesion scores of its children or its grand-children. However, it depends indirectly on the children's scores because they have been chosen for their optimality. Therefore the algorithm goes through all the possible configurations of children constituents in order to determine which of these configurations is optimal. The algorithm is not required to also consider the grand-children, since their influence only factors through the immediate descendants of the constituent being generated.

□

§4.8 discusses how to use the cohesion of constituents in order to prune the search space.

4.6 Complexity

Theorem 4.1 (Complexity of Loose Satisfaction Chart Parsing) *The solution proposed for LSCP through Algorithm 2 to Algorithm 4 presents a worst time complexity of*

$$\mathcal{O}(n^4 2^n C^n)$$

where C is the number of constructions specified in the grammar, and n the input size in words.

Proof The full algorithm is made up of 5 main loops, 4 of which are visible in Algorithm 2, the fifth one being the main loop of the `characterisation`() function. We label the loops (L1) to (L5), (L1) referring to the top-level one, and (L5) to the most embedded one.

1. (L1) loops over the input size, thus iterates n times.
2. (L2) iterates at most n times. More precisely, (L2) iterates over the span offset, and at every iteration from (L1), (L2) iterates one time less than previously. Therefore, combining (L1) and (L2) represents

   $$\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$$

   iterations.
3. (L3) enumerates the $2^{\text{span}} - 1$ partitions of the current window. With $span \leqslant n$ we can safely take

   $$2^n$$

as an upper bound for (L3).

4. (L4) iterates over all the configurations for the current partition. Let p be the current partition of the current span such that $p = \{\{p_i\}_{1 \leqslant i \leqslant s}\}$, where p_i is a k-subset and $s = \mathrm{card}(p)$ is the number of k-subsets. let k_i be the set of distinct constituents stored in π for p_i, with $c_i = \mathrm{card}(k_i)$. We know that each p_i is stored in a different cell of the chart. The configurations are obtained by combining every constituent found in π for one p_i with every constituents of every other p_i. Therefore, the total number of configurations for p is

   $$\delta_p = \prod_{i=1}^{s} c_i$$

   Given that $s = \mathrm{card}(p) \leqslant \mathit{span}$, we can take n as an upper bound of s. If we take the total number C of constructions in the grammar as an upper bound of c_i, then an upper bound for δ_p is

   $$C^n$$

5. The characterisation of a configuration sees a configuration as an assignment for the grammar, where the grammar is a constraint system. The characterisation task involves looking up the constraints in the grammar. We have seen in §4.4.5 that it is done by consulting a lookup table, where the constraints are indexed by their variables being either free or universally quantified (hereafter the *keys*). Therefore we need to enumerate all the combinations of variables, which may be keys. The variables are the immediate constituents of the current configuration. Each variable is of one and only one construction. By extension, and for the sake of the calculation, we assimilate here a variable to its construction.

   Let p be a configuration of m constructions, and let k be the maximum cardinality of a key in the grammar, in number of constructions. The number of combinations for p is

   $$\delta_p = \sum_{i=1}^{k} \binom{m}{i}$$

The size of k depends on the semantics of the constraint types. Currently, there exists no constraint type defined with a $k > 2$. Therefore, we have

$$\delta_p = \binom{m}{1} + \binom{m}{2} = m + \frac{m(m-1)}{2} = \frac{1}{2}m(m+1)$$

If we take n as an upper bound for m, we obtain

$$\frac{1}{2}n(n+1)$$

as an upper bound of the number of lookup table consultations.

6. (L5) iterates over all the constituents inferred by an assignment. Therefore a (rather large) upper bound for the number of iterations is the number C of constructions specified in the grammar.

7. (L5) being embedded in (L4) at the same level as the characterisation process, what we want in fact is the max of 5. and 6. above, *i.e.* $M = \max(\frac{1}{2}n(n+1), C)$. M is determined by the sign of $n(n+1) - C$; thus for $n \geq \sqrt{C}$ we have $M = \frac{1}{2}n(n+1)$. Thus for a grammar such as the on presented in Appendix A, which counts 16 constructions, n just need to be greater than 3.

8. In the end, we have

   for (L1) and (L2) combined: $\frac{1}{2}n(n+1)$

   for (L3): 2^n

   for (L4): C^n

   for (L5) and the characterisation: $\frac{1}{2}n(n+1)$, for $n > 3$

In conclusion, the complexity of Algorithm 2 is

$$\begin{aligned}\frac{1}{2}n(n+1) \cdot 2^n \cdot C^n \cdot \frac{1}{2}n(n+1) &= \frac{1}{4}n^2(n+1)^2 2^n C^n \\ &\simeq n^4 2^n C^n\end{aligned}$$

□

4.7 The Corpus of Acceptability Judgements

Before we go any further we need, for the sections to come, to briefly introduce the corpus developed by Blache, Hemforth, and Rauzy (2006). It will be described in more details in §5.2.

This corpus was artificially constructed, in that it results from systematically applying a list of pre-defined patterns of errors to base sentences, in order to create new ones. The sentences are annotated with numeric human judgements of acceptability, as a result of psycholinguistic experiment with syntactic gradience. For what we are concerned in this chapter the corpus has the important advantage, in the context of evaluating *Numbat*, of being essentially made up of deviant language, which allow us concentrate on evaluating the robustness of *Numbat*. Because built systematically, it especially has the advantage of enabling us focus on the performance with respect to a list of specific error patterns. These patterns, reproduced in Table 4.4, were controlled by the investigators during the creation of the sentences in the corpus.

Meanwhile, the corpus also has various disadvantages. One of them is its small size (about 12,000 words in 1,000 sentences), compared to the corpora in use for evaluating large-scale parsers. This being said, it has never been our intention in this work to compete with these parsers performance-wise[15]. In any case, statistical parsers do not, and can not, provide us with the characterisation of an input which let us model gradience.

Another disadvantage is that the range of constructions being covered in the corpus is quite limited—not only in terms of the number of error patterns, but also in terms of the number of contexts in which these patterns are applied. The complexity of the constructions involved in the base sentences is not very high, in that no sentence is more than 15 words long, none contains embedded clauses such as coordinate, or subordinate clauses for instance, none presents a cleft structure, and so on.

[15] However, the performance of a statistical parser is provided as a baseline for comparison with *Numbat*, but on a different corpus made of plain text. See §4.9.2 for details.

No violations		
1.1	Marie a emprunté un très long chemin pour le retour *Marie* [aux.] *followed a very long path on the way back*	
NP-violations		
2.1	Marie a emprunté très long chemin un pour le retour *Marie* [aux.] *followed very long path a on the way back*	
2.2	Marie a emprunté un très long chemin chemin pour le retour *Marie* [aux.] *followed a very long path path on the way back*	
2.3	Marie a emprunté un très long pour le retour *Marie* [aux.] *followed a very long on the way back*	
2.4	Marie a emprunté très long chemin pour le retour *Marie* [aux.] *followed very long path on the way back*	
2.5	Marie a emprunté un très heureux chemin pour le retour *Marie* [aux.] *followed a very happy path on the way back*	<
AP-violations		
3.1	Marie a emprunté un long très chemin pour le retour *Marie* [aux.] *followed a long very path on the way back*	
3.2	Marie a emprunté un très long long chemin pour le retour *Marie* [aux.] *followed a very long long path on the way back*	
3.3	Marie a emprunté un très chemin pour le retour *Marie* [aux.] *followed a very path on the way back*	
3.4	Marie a emprunté un grossièrement long chemin pour le retour *Marie* [aux.] *followed a roughly/rudely long path on the way back*	<
PP-violations		
4.1	Marie a emprunté un très long chemin le retour pour *Marie* [aux.] *followed a very long path the way back on*	
4.2	Marie a emprunté un très long chemin pour pour le retour *Marie* [aux.] *followed a very long path on on the way back*	
4.3	Marie a emprunté un très long chemin le retour *Marie* [aux.] *followed a very long path the way back*	
4.4	Marie a emprunté un très long chemin pour *Marie* [aux.] *followed a very long path on*	
4.5	Marie a emprunté un très long chemin dans le retour *Marie* [aux.] *followed a very long path in the way back*	<
VP-violations		
5.1	Marie un très long chemin a emprunté pour le retour *Marie a very long path* [aux.] *followed on the way back*	
5.2	Marie a emprunté emprunté un très long chemin pour le retour *Marie* [aux.] *followed followed a very long path on the way back*	
5.3	Marie un très long chemin pour le retour *Marie a very long path on the way back*	
5.4	Marie emprunté un très long chemin pour le retour *Marie followed a very long path on the way back*	
5.5	Marie a persuadé un très long chemin pour le retour *Marie* [aux.] *convinced a very long path on the way back*	<

Table 4.4: Error patterns. *The left-most column contains a numbered label for the pattern type and sub-type; sentences marked with < are removed from the corpus, for reasons explained in due course in Chapter 5.*

4.8 Heuristic

Given its complexity, implementing Algorithm 2 requires resorting to practical heuristics in order to prune the search space drastically. We choose to use a single one[16], which has a limited impact on the correctness of the general algorithm. Following VanRullen (2005), constituents are pruned on the basis of the proportion of constraints they strictly satisfy, with respect to the full number of constraints which either hold or fail. That is, they are pruned according to their cohesion[17] (see §4.4.4). A user-defined *satisfaction threshold* determines how loose the parsing process may be and subsequently how loose a constituent structure may be. Note that this threshold does not aim to address the cases grammatical ambiguity inherent to a natural language. Whether it could—or not—be used to serve that purpose as well is left open for further investigation. As far as *Numbat* is concerned in such a case the user can choose whether the outcome should be all the alternative parses or just one, picked randomly.

4.8.1 Fixing a Satisfaction Threshold

In order to fix a satisfaction threshold two questions must be addressed:

1. (**Scope**) What scope does it have? That is, which level in the parsing process does it apply to?

2. (**Value**) What value does it have, and how to determine it?

Scope

There are different places in the process where the heuristic can be applied. These places are not necessarily contradictory. They may be combined strategically and

[16] Another heuristic probably worthwhile for efficiency purpose would be to bound the number of constraints. Something inspired from probabilistic approaches might help: at the moment, all the configurations are taken into account, even though most of them could be deemed irrelevant, because very unlikely. We believe that these configurations could be pruned using something like n-grams probabilities, for example. However, such optimisation should be envisaged for practical efficiency, but is incompatible with showing the optimality of the solution parse (the most probable parse not being necessarily the same as the optimal one).

[17] A different heuristic could have been based on some mean value taking into account both the cohesion and a dual of it.

consistently. Each place is interpreted differently and corresponds to a strategic orientation given to the process.

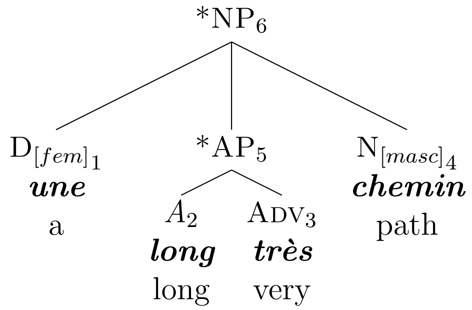
General Threshold A satisfaction threshold may range over the entire process, and fix that no constituent whatsoever may be characterised by less than a ratio of $\theta \cdot 100\%$ of satisfied constraints. For instance, in fixing $\theta = 1.0$ the parser behaves like a strict parser, ruling out any ill-formed constituent. When $\theta < 1.0$, loose constraint satisfaction applies, and some degree of robustness is achieved by allowing loose constituents, whose characterisation may be made of up to $(1 - \theta)\%$ of violated constraints.

The problem with such a general threshold is that it has only got the strength of the weakest link in the grammar: the number of constraints characterising a constituent shows significant variations across the grammar, some constituents being characterised by a much larger number of constraints than others. The smaller the characterisation, the more important, in proportion, the violation of one constraint. Example 4.9 illustrates such a situation. Subsequently, in order for the threshold to let in *poor constituents*—*i.e.*, whose characterisation concerns a small number of constraints, it must be set quite low. And if set to a low value, then more *rich* constituents are allowed to compete as candidate structures, which increases the complexity and somehow inhibits the impact of the heuristic.

Example 4.9 (*Rich* vs. *Poor* Constituents) *The deviant Noun Phrase* une long très chemin *in French (*a long very path*) violates:*

- *a gender agreement between determiner (D) (feminine) and noun (N) (masculine), and*
- *word order, between adverb (Adv) and adjective (A)*

The constituent structure generated by Numbat is the following:



Let us have a look at the characterisations of AP_5 *and* NP_6 *with respect to* Γ. *With* $\mathcal{A}_5 = \langle \mathrm{A}_2, \textsc{Adv}_3 \rangle$ *the following constraints are loosely satisfied:*

$$\mathcal{A}_5 \overset{\frown}{\models} \text{(P3.30)} \wedge \text{(P3.31)} \wedge \text{(P3.33)} \wedge \text{(P3.36)} \wedge \text{(P3.35)} \tag{4.17}$$

(4.17) corresponds to the following characterisation:

$$\chi_{\mathcal{A}_5} = \begin{cases} \chi^{+}_{\mathcal{A}_5} = \text{(P3.30)} \wedge \text{(P3.31)} \wedge \text{(P3.33)} \wedge \text{(P3.36)} \\ \chi^{-}_{\mathcal{A}_5} = \text{(P3.35)} \end{cases} \tag{4.18}$$

All the constraints in $\chi_{\mathcal{A}_5}$ *project the construction* AP.
Now, with $\mathcal{A}_6 = \langle \mathrm{D}_1, \mathrm{AP}_5, \mathrm{N}_4 \rangle$ *the following constraints are loosely satisfied:*

$$\begin{aligned} \mathcal{A}_6 \overset{\frown}{\models} & \text{(P3.6)} \wedge \text{(P3.7)} \wedge \text{(P3.8)} \wedge \text{(P3.11)} \wedge \text{(P3.13)} \\ & \wedge \text{(P3.15)} \wedge \text{(P3.16)} \wedge \text{(P3.17)} \wedge \text{(P3.18)} \wedge \text{(P3.18)} \end{aligned} \tag{4.19}$$

(4.19) corresponds to the following characterisation:

$$\chi_{\mathcal{A}_6} = \begin{cases} \chi^{+}_{\mathcal{A}_6} & = & \text{(P3.6)} \wedge \text{(P3.7)} \wedge \text{(P3.8)} \wedge \text{(P3.11)} \wedge \text{(P3.13)} \\ & & \wedge\text{(P3.15)} \wedge \text{(P3.16)} \wedge \text{(P3.17)} \wedge \text{(P3.18)} \\ \chi^{-}_{\mathcal{A}_6} & = & \text{(P3.18)} \end{cases} \tag{4.20}$$

In (4.18) the constraint (P3.35) *represents* 20% *of failure (1 constraint out of 5), while in (4.20) the constraint* (P3.18) *represents only* 10% *of failure (1 out of 10). Therefore, the* AP *is a poor constituent compared to* NP.

A solution to that problem is to reduce the scope of the threshold to the construction level.

Construction-specific Threshold A satisfaction threshold may also apply to a specific construction. The main reason for it is to locally attenuate the effect of the general threshold. That is, it attenuates the fact that a poor constituent being characterised with a very small number of constraints in the grammar may be pruned much faster than a rich constituent being characterised with a large number of constraints (*e.g.* AP *vs.* VP, as in Example 4.9).

In *Numbat*, we decided to use both a general threshold as a default one, and construction-specific thresholds in order to lower or raise the general one, according to cases.

In Example 4.9, the AP construction requires a lower threshold than NP in order for loose constituents to be licensed.

Error Pattern Threshold A third type of threshold we experimented with in *Numbat* applies to the different error patterns present in the corpus we used for experimenting with syntactic gradience (see §5.2 in Chapter 5 for a description of these patterns). This threshold allows a finer granularity of the general threshold and is only relevant to our corpus. It was mostly used during the grammar-writing phase in order to find out, in cases where the main solution was incorrect, whether a better structure was proposed among the alternatives and could be reached by lowering a threshold. This way, the main threshold could be lowered only for the sentences from one specific type (*i.e.* by error pattern), without impacting too much the processing time—as it would be the case if the main threshold itself was lowered. The different values used by sentence type is presented in Table 4.5.

In practice, for each construction the ruling threshold was the lowest of the three.

Pattern	1	2.1	2.2	2.3	2.4	2.5	3.1	3.2	3.3	3.4
Threshold	1.0	0.8	0.9	0.7	0.8	N/A	0.8	0.8	0.8	N/A
Pattern	4.1	4.2	4.3	4.4	4.5	5.1	5.2	5.3	5.4	5.5
Threshold	0.8	0.8	0.8	0.8	N/A	0.8	0.8	0.8	0.8	N/A

Table 4.5: Values for the Threshold by Error Pattern. *See §5.2 in Chapter 5 for a description of these patterns.*

Scope	General	S	VP	NP	PP	AP	Rel	Coord	Compl
Value	0.8	0.7	0.8	0.8	0.7	0.5	0.9	1	0.9

Table 4.6: Values for the Construction-specific Threshold

Value

In order to decide on an optimal value for the construction-specific threshold for each construction, we ran *Numbat* on the same corpus as the one we are using in Chapter 5 (see §5.2 for a detailed description) until we reached a reasonable compromise between processing time and proportion of full parses in the outcome. The corpus we used is essentially made up of deviant utterances.

Ultimately, we obtained the values presented in Table 4.6.

4.9 Evaluation

It is virtually impossible, when evaluating a parser, to properly and completely discriminate evaluation of the grammar from evaluation of the parsing strategy itself. Our evaluation of *Numbat* is no exception. However, elements of answer can be found in interpreting different observations we make about the parser's outcome and its numerical evaluation. These elements help understanding the mutual influence of grammar and parsing strategy. In this section, we discuss the most important of these elements and draw conclusions about the grammar, LSCP as a parsing strategy, and the *Numbat* parser as its implementation.

4.9.1 What and How to Evaluate?

Because the main goal that *Numbat* is trying to achieve is to generate full parses not only for grammatically well-formed sentences but also for grammatically ill-formed ones (quasi-expressions), we would like to evaluate its performance on both aspects, which does not make the task straitforward. In the following we present two separate evaluations: a first one, which aims to evaluate the performance on well-formed input and to compare it with other existing parsers, and a second one, which aims to evaluate the performance on ill-formed input and to measure to what extent it provides a reliable input to the model of gradience—*i.e.* fully parsed quasi-expressions.

As far as well-formedness is concerned, evaluating a parser and the quality of its output usually consists of measuring how it performs at parsing a corpus for which an annotated reference exists—in our case a treebank. Since we are working with French the only such treebank currently available is the one developed by Abeillé, Clément, and Toussenel (2003). However, running an evaluation using this treebank was made quite difficult mainly because of a lack of resources. In general terms, the use of a treebank as Gold Standard requires different resources to be available and suitable to the annotation scheme adopted in it. In particular, it is critical that the dictionary used by the parser to be evaluated rely on the exact same tagset as the one used to annotate the Gold Standard, in order for the POS-tagging to be appropriate. As far as we know such a dictionary is not publicly available.

The grammar is also at stake, and requires to be developed following the same annotation guide as the one provided to the human annotators for creating the reference. Whether developed manually or automatically—we briefly discuss in §5.5.4 the possibility of deriving such a grammar automatically from an existing one and the problems that it raises—the task is labour-intensive and could not be envisaged in the time frame of this study.

Hence the alternative to Abeillé et al.'s treebank which we opted for, which is to evaluate *Numbat* according to the protocol designed for the French competition-based evaluation EASY, since we have got all the resources available. This evaluation is detailed and discussed in §4.9.2.

As for ill-formedness, the range of possibilities for evaluating full parses is even narrower. We opted for asking expert human annotators to make binary judgements about the *Numbat*'s output parse trees. This evaluation is detailed and discussed in §4.9.3.

4.9.2 Evaluation ♯1: EASY

EASY (Évaluation d'Analyseurs SYntaxiques) is a competitive evaluation campaign for parsers of the French language, which took place in 2004–2005 (Gendner et al., 2003; Vilnat et al., 2003, 2004) and involved 15 participants. The annotation guide (Gendner et al., 2003; Paroubek, Robba, and Vilnat, 2003) includes an example-based description of how constituents are expected to be analysed in phrases (*groupes*). A specific set of 6 phrase types was designed for the purpose, where a constituent does not include any nested children constituents other than Part-of-Speeches, and is not embedded either in any dominant constituent structure. The resulting structure for a sentence is flat, without any unique root. The 6 phrase types are:

(i) Noun Phrase (GN)

(ii) Adjective Phrase (GA)

(iii) Preposition Phrase (GP)

(iv) Adverb Phrase (GR)

(v) Verb Phrase (NV)

(vi) Preposition-Verb Phrase—for Verb Phrases introduced by a preposition (PV)

The evaluation uses crossing bracket plus precision and recall measures. Relations could also be evaluated separately, but Numbat does not generate any thus we do not tackle that aspect of EASY here. During the competition the precision and recall measures were applied 15 different constraint relaxations (Paroubek, Robba, and Ayache, 2007), allowing to loosen up to different degrees the constituent boundaries and the constituent specifications. Since it is unclear which of these constraints were applied for gathering the reported results, and since some specific implementation details were also unclear such as how the punctuation marks were accounted for, we could not ensure that the exact same measurements were replicated. In the evaluation we have run[18] we assume the hardest constraints, where both left and right boundaries must be the same as the reference, and the constituent's label must be the same as well. Because using the published results as a baseline for comparison with *Numbat*'s performance would potentially be inaccurate for the reasons mentioned above, we have run the same evaluation for two other parsers developed at Université de Provence and we compare their performance to that of *Numbat*. One is a shallow parser, which was involved in the EASY competition, and the other one is a stochastic parser.

The Corpus The one-million word corpus to be parsed is made up of different sources: *general*, containing "Le Monde" newspapers, and meeting reports from the French Senate, among others; *literature*; *e-mails*; *questions* from the TREC conference and the Amaryllis project; transcribed *spoken* language; and *medical* publications. The distribution in type of source is the following:

general	21%
literature	23%
e-mail	15%
medical	6%
spoken	28%
questions	7%

The Gold Standard in use for the evaluation is a 10% sample of the main corpus, annotated manually.

The Grammar The grammar we use was developed by Balfourier et al. (2005), and participated in the 2004–2005 campaign with VanRullen's SeedParser. It is made up of 8 POS (listed in Table 3.1) plus the 6 EASY categories, thus amounting to 14 constructions. A total of 185 constraints specify all of them. The full specifications of the grammar are reported in Appendix B.

[18] Credit goes to Stéphane Rauzy from LPL, CNRS (Université de Provence), for implementing an evaluation programme on the basis of all the information available regarding the EASY measurements.

The Lexicon The lexicon we use, DicoLPL, was also developed by VanRullen et al. (2005) (see also VanRullen, 2005, Chapter 9 for a detailed evaluation). DicoLPL contains 444,000 forms, and was evaluated on a 153-million corpus of French newspapers. Interestingly, the 54,000 most frequent forms are reported to cover 99% the corpus, and 10,000 forms only are sufficient to cover 90% of it.
The POS-Tagger we use was developed by VanRullen (2005) as part of the LPLSuite.

Results Table 4.7 reports the results of the evaluation we performed of *Numbat*'s output with Balfourier et al.'s EASY grammar. In order to be comparable with our experiments with gradience later on, and because of the rather poor performances

	Precision	Recall	F-measure
Total	**0.7835**	**0.7057**	**0.7416**
general_lemonde	0.8187	0.7515	0.7837
general_mlcc	0.7175	0.6366	0.6746
general_senat	0.8647	0.7069	0.7779
litteraire_1	0.8331	0.7734	0.8022
litteraire_2	0.8413	0.8103	0.8255
litteraire_3	0.805	0.7527	0.778
litteraire_4	0.7702	0.724	0.7464
mail_10	0.7366	0.7154	0.7258
mail_9	0.702	0.6749	0.6882
medical_2	0.8712	0.7	0.7763
medical_3	0.8393	0.6839	0.7537
medical_6	0.8614	0.65	0.7409
oral_delic_4	0.8494	0.7051	0.7705
oral_delic_5	0.5827	0.5	0.5382
oral_delic_6	0.6475	0.5676	0.6049
oral_delic_7	0.6389	0.5833	0.6098
oral_delic_8	0.7885	0.8077	0.798
oral_delic_9	0.5833	0.5625	0.5727
questions_amaryllis	0.8081	0.7432	0.7743
questions_trec	0.8208	0.7069	0.7596

Table 4.7: Evaluation of *Numbat* according to the EASY Protocol

of *Numbat* time-wise and memory-wise[19], we look at a subset of the EASY corpus, which is made up of all the sentences of a maximum of 15 words—15 words being the maximum sentence length in the other corpus which we are using for experimenting with gradience. For comparison purpose Tables 4.8 and 4.9 report the results obtained by respectively a shallow parser (*ShP*) using a left-corner strategy, and a stochastic parser (*StP*), which were developed by Blache and Rauzy (2008). Both of them were developed at Université de Provence and were involved in the EASY competition.

	Precision	Recall	F-measure
Total	**0.7846**	**0.8376**	**0.8102**
general_lemonde	0.806	0.8569	0.8306
general_mlcc	0.8118	0.8678	0.8389
general_senat	0.8228	0.867	0.8443
litteraire_1	0.7933	0.84116	0.8165
litteraire_2	0.7948	0.8543	0.8235
litteraire_3	0.7846	0.8297	0.8065
litteraire_4	0.7622	0.8306	0.795
mail_10	0.7341	0.7695	0.7513
mail_9	0.7260	0.7724	0.7485
medical_2	0.8389	0.8646	0.8515
medical_3	0.8093	0.8569	0.8324
medical_6	0.8378	0.8748	0.856
oral_delic_4	0.7275	0.7442	0.7358
oral_delic_5	0.7736	0.82	0.7961
oral_delic_6	0.6270	0.7577	0.6862
oral_delic_7	0.6138	0.7249	0.6648
oral_delic_8	0.7662	0.8270	0.7955
oral_delic_9	0.705	0.7899	0.7450
questions_amaryllis	0.8334	0.8714	0.8522
questions_trec	0.7723	0.8329	0.8014

Table 4.8: Evaluation of the shallow parser *ShP* according to the EASY Protocol

Table 4.10 reports the details of the cross bracket scores. The apparent discrepancies observed between the total scores in Table 4.7 and Table 4.10 comes from that in Table 4.10 the total scores are macro-averages of the constituent scores, where the

[19] It takes *Numbat* about 30 hours to process the subset of the 15-word sentences of the EASY corpus on a single machine—a two-CPU Intel Xeon® @ 2.6 MHz with 4GHz memory.

	Precision	Recall	F-measure
Total	**0.9013**	**0.8978**	**0.8995**
general_lemonde	0.9221	0.9205	0.9213
general_mlcc	0.9224	0.9146	0.9185
general_senat	0.9240	0.9178	0.9209
litteraire_1	0.9261	0.9213	0.9237
litteraire_2	0.9128	0.9252	0.9190
litteraire_3	0.8999	0.8962	0.8980
litteraire_4	0.9139	0.9159	0.9149
mail_10	0.8322	0.8110	0.8215
mail_9	0.8625	0.8462	0.8543
medical_2	0.8885	0.8854	0.8869
medical_3	0.9005	0.9004	0.9004
medical_6	0.9282	0.9163	0.9222
oral_delic_4	0.8361	0.8342	0.8352
oral_delic_5	0.9003	0.8567	0.8779
oral_delic_6	0.7905	0.8145	0.8023
oral_delic_7	0.7692	0.7321	0.7502
oral_delic_8	0.8759	0.8829	0.8794
oral_delic_9	0.8248	0.8675	0.8456
questions_amaryllis	0.9210	0.9242	0.9226
questions_trec	0.9117	0.8967	0.9042

Table 4.9: Evaluation of the stochastic parser *StP* according to the EASY Protocol

weight for each constituent is determined by $\frac{\text{Number of constituents observed for that category}}{\text{Total number of constituents observed}}$.

	Precision	Recall	F-measure
Total	**0,7670**	**0,7057**	**0,7350**
`<GN>`	0,7639	0,7942	0,7788
`</GN>`	0,6736	0,7004	0,6867
`<GP>`	0,6952	0,6024	0,6455
`</GP>`	0,8227	0,7129	0,7639
`<NV>`	0,8209	0,7418	0,7794
`</NV>`	0,7996	0,7225	0,7591
`<GA>`	0,8491	0,6870	0,7595
`</GA>`	0,7642	0,6183	0,6835
`<PV>`	0,8155	0,7534	0,7832
`</PV>`	0,8301	0,7668	0,7972
`<GR>`	0,8354	0,6947	0,7586
`</GR>`	0,7595	0,6316	0,6897

Table 4.10: Cross bracket measures of *Numbat*'s output by constituent type, according to the EASY Protocol

Interpretation In achieving a precision of 0.7835 *Numbat* is nearly as good as the shallow parser *ShP* and its 0.7846 precision score, and is (expectedly) outperformed by the stochastic parser *StP* which scores a 0.901 precision. However, on recall *Numbat* is significantly weaker than both *ShP* (0.706 *vs.* 0.838), and *StP* (0.898).

An interesting first conclusion that can be drawn from these figures is that it emphasises *Numbat*'s ability to adapt to different grammars. In that case, the EASY grammar is significantly different from the one we use for experimenting with gradience in terms of output structure: the EASY output is a flat structure, where the top-level constituents are only made up of Part-Of-Speech, whereas the grammar involved in our experiments is designed for full hierarchical constituent structures, with nested constituents. As far as *Numbat* is concerned, it must be acknowledged that the version of LSCP it implements was more focused on producing nested structures than flat ones. Despite such a difference in terms of expected outcome, *Numbat* is flexible enough to handle it with a precision similar to the one of a shallow parser.

As for recall, the score can be explained by looking at one of the weakest link in the parsing process when it comes to generate an EASY outcome. An element of explanation comes indeed from the choice of heuristic that we have made for choosing a preferred constituent that has no dominant category. we have introduced that numeric heuristic (the *preference score*) in §4.4.3, where we detail the Memoization Step of LSCP. As a reminder, the preference score is different from the cohesion score (defined in §4.4.4); its purpose is merely to continuously maintain a preferred configuration of constituents should a cell in the chart not being populated, since in that case the cohesion can not be calculated. This preferred configuration is always dominated by a root constituent of the wildcard type *Star*. In the case of an EASY grammar, where all the specified constructions are flat, we fall into that case and the choice of the (optimal) output structure essentially relies on this heuristic. For memory, the preference goes to constituents with the widest span and the least overall number of constituents. This translates in the following expression:

$$p_F = span \cdot (\texttt{merit}(C_i) + span)$$

At this stage the preference score is simply used as a heuristic, and is only one of the many possibilities that exist for expressing it. Clearly, there might be better numeric functions, for instance to suit the EASY structure and achieve a better recall, that would be worth investigate. We leave that option open for further works.

The reason why computing the cohesion is not possible is that parsing according to the EASY grammar is in fact similar to the kind of situations that leads, in the general case, to output a forest of partial parses: LSCP, through its CKY basis, is primarily designed to produce a full parse, and is not optimised to produce a flat structure. Initially, one might expect to see a flat structure spanning the input string as a set of nodes belonging to the same virtual parse tree, which would connect all of them to each other. A naive approach might then be to simply stop the parsing process as soon as all the nodes we need were constructed. However, such an approach is, in fact, not compatible with LSCP because the process is not incremental, which here means that the final decision about which constituents belong the solution is postponed until a decision can be made about the optimality of the full structure. As a consequence, it is necessary to attempt building a full parse in order to know how to group together the words from the input string. In other words, it is necessary to complete the entire chart before a structure that spans the entire input, even a flat structure, can be decided.

4.9.3 Evaluation ♯2: quasi-expressions

In that evaluation we asked five independent annotators, all linguists, to assess whether the *Numbat*'s output parse trees were *correct* or not. The corpus in use was the one introduced in §4.7, made up of 94% of quasi-expressions. The annotators were asked (see Appendix C for the full instructions, in French) to answer the following question for every parse tree presented:

> *In your opinion, does the syntactic tree associated with each sentence constitute a possible and acceptable parse for the input sentence?*
>
> It is important to emphasise that the acceptability judgement must be performed, as much as possible, on the basis of *syntactic* criteria only. The objective is not to make a judgement about the acceptability of the sentence as such (most of them are intentionally ill-formed) but rather about the syntactic parse provided for it. The difficulty stands in the assessment of the adequation between ill-formedness and full syntactic parse.

The possible answer was necessarily binary: the parse structure is correct or not. Further instructions were given to ensure that the acceptability judgement was based, as much as possible, on syntactic criteria only, and that acceptability was concerning the parse tree, rather than the sentence itself.

We acknowledge that the reliability of such an evaluation is arguable. A possible improvement would be to run a more thorough psycholinguistic experiment, similar to the one that is available for sentence graded acceptability (see Chapter 5, especially §5.2), in order to assess whether the syntactic parse assigned to an ill-formed sentence is “sufficiently good” or whether it is “bad”. Yet, since these automatically generated parses are taken as input for the different rating schemes that we discuss in Chapter 5, we considered it was important to have at least an idea of “how good” these *full* parses are, hence this evaluation. We insist on *full* parses, in order to stress that the EASY evaluation protocol is not enough in this respect, as we pointed out in §4.9.2.

Note that we did not ask the annotators to assess whether the parse was the *best* possible one, but simply whether it was "a possible and acceptable [one]", deciding on the best possible one would have made the task harder and probably less reliable. The objective is essentially to determine whether the parse which is considered as the best one by the parser constitutes a reasonable input for the numeric models of gradience by human judgements.

Note, as well, that a similar argument of reliability could apply to §4.9.4 where we discuss elements of accuracy and aim to interpret why a syntactic parse is considered as an error: rather than relying only on our own intuition in order to determine whether a parse was an error, this evaluation, to some extent, helps to support the discussion.

Grammar and Lexicon The lexicon we use is the same as the one used for the EASY evaluation and reported in §4.9.2. The grammar (PGCxG) was already briefly mentioned earlier in §3.2.4, where a sample of it is presented as the toy grammar Γ. PGCxG contains 16 constructions, including 8 POS, and 100 constraints are required to specify it. Besides the 8 POS and the other 5 constructions presented in 3.1 (4 phrasal constructions plus sentence S) the remaining 3 constructions are:

- Relative clause
- Completive clause
- Conjunction clause

The full specifications of PGCxG can be found in Appendix A.

The Measurements Since, to the best of our knowledge, there exists no established evaluation procedure in the context we have set, we propose to adapt the *precision* and *recall* measurements. Our purpose is to provide measures which account for how many of the solutions are full parses—as opposed to forests of partial parses, and

User	Correct	Total user	Complete parses	Precision = $\frac{Correct}{Complete}$	Recall = $\frac{Correct}{Total}$
1	77	112	101	0.76	0.69
2	79	112	101	0.78	0.71
3	88	112	105	0.84	0.79
4	139	231	211	0.66	0.6
5	86	127	114	0.75	0.68
TOTAL	469	694	632	**0.74**	**0.68**

Table 4.11: Precision and Recall measures for Evaluation ♯2 on quasi-expressions

how many of these full parses are correct by human judgement. In the absence of a reference annotation we propose to interpret:

- the total number of input sentences as the number of *predictions*,
- the number of COMPLETE parses as the number of *observations*,
- the number of CORRECT parses by human judgement as the number of correct solutions.

Thus we obtain:

$$\text{precision} = \frac{\text{CORRECT}}{\text{COMPLETE}}$$

$$\text{recall} = \frac{\text{CORRECT}}{\text{total}}$$

Results The results presented in Table 4.11 show a precision of 0.74, and a recall of 0.68. It gives an F-measure of 0.71:

$$F = 2 \cdot \text{precision} \cdot \text{recall}/(\text{precision} + \text{recall}) \simeq 0.71$$

The full parses represent 92% of the sentences in the corpus. The measure of precision indicates that 74% of the full parses are evaluated as syntactically correct, while the recall indicates that the correct full parses represent 68% of the entire corpus. Considering that the quasi-expressions represent 94% of the entire corpus, and when compared with the scores on well-formed sentences (precision/recall/F-measure = 0.78/0.71/0.74), these measures show a rather good achievement of *Numbat* over deviant language.

4.9.4 Elements of Accuracy

The measures of precision and recall show the extent to which there is agreement across the results, and the measures fail to inform on different aspects of accuracy. Unfortunately, providing overall measures of accuracy is not straightforward. It is nevertheless possible to provide cues about specific phenomena that are observed.

The two questions about accuracy we are mostly interested in answering are:

1. Given the possibly deviant syntactic characteristics of a string (*i.e.*, its characterisation), is this string categorised in the correct construction?

2. Is a deviant (sub-)string actually found deviant by *Numbat*?

We focus on these two questions because they are the most important ones in order to evaluate the quality of the syntactic information our experimental models of gradience will be tested on. As we already discussed it in Chapter 2, an assessment of the degree of grammaticality of an item requires the knowledge of two elements in order to be performed:

1. a class—in our case, a construction; that is the class with respect to which the degree of grammaticality is calculated;

2. the item's (syntactic) characteristics.

Regarding the latter, and since we are concerned with deviant language, we want to make sure that broken characteristics are correctly identified, since they will play a major role in calculating a degree.

Although these two questions are seemingly quite typical of a categorisation problem, they turn out to be less than trivial to evaluate when aspects of gradience enter the picture. Yet different phenomena are clearly observed, and the systematic classification of the utterances in the corpus into types and sub-types—according to error patterns applied—allow us to evaluate, to some extent, how these phenomena are dispersed across the corpus. Interestingly, Evaluation ♯2 also reveals a significant collection of unexpected cases, in spite of all the care put by the investigator into controlling every aspect of the corpus. We discuss here the most salient of these cases.

Categorisation Failure

Case 1: Missing Adjectives In the case exemplified in Figure 4.2 the adverb *très* (*very*) is seen by *Numbat* as modifying the Verb, whereas it is expected to be modifying a (missing) Adjective. The bracketing is incorrect, since the Adverb belongs here to the VP instead of being part of the NP. The parser fails to categorise the Adverb correctly. These cases are evaluated by annotators as syntactically incorrect. The problem here comes from a drawback in POS-tagging. A morphological difference ought to be made between an adverb modifying an adjective, and an adverb modifying

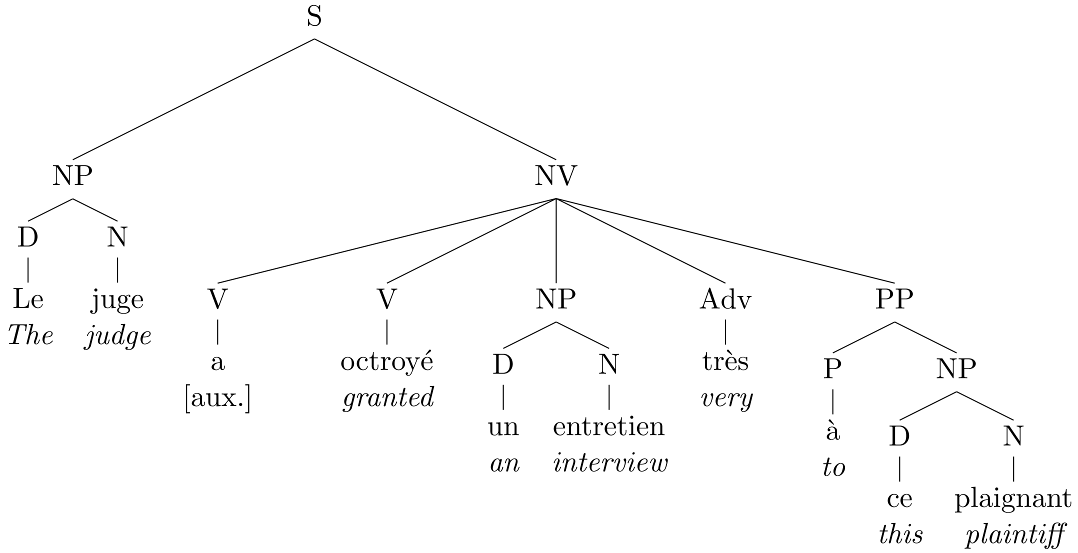


Figure 4.2: *Numbat*'s output for case of missing Adjective—*The judge* [aux.] *granted a very interview to this plaintiff. The Adverb is analysed as a modifier of the Verb, whereas it is expected to be modifying a (missing) Noun.*

a verb[20]. Unfortunately, the POS-tagger we are using (VanRullen, 2005) does not make such a difference.

Characterisation Failure

Case 2: Substantive Adjectives In the case exemplified in Figures 4.3 and 4.4 the syntactic structure provided by *Numbat* is evaluated as *correct* by annotators, whereas it is meant to be deviant, since it results from an error pattern. The problem is that *Numbat* analyses an Adjective Phrase as being head of a Noun Phrase, whereas this AP was meant to be modifying a (missing) Noun. This case reveals an under-specification of the grammar. Indeed an Adjective Phrase head of a Noun Phrase is licensed by the grammar, in order to cover cases of substantive adjectives. Therefore, sentences such as those in Figures 4.3 and 4.4, where an adjective does not modify any Noun, are identified as cases of substantive adjectives by the parser, whereas they are not. This case is a potentially troublesome because it is rightfully structured, but for the wrong reason. The missing Noun is expected not to prevent identifying an NP, and indeed it does not. However, no constraints are found to fail, which is expected to entail a wrong assessment of the utterance's gradience. The parser fails to find the case deviant, although it is actually so.

The phenomenon reveals a case of under-specification, but it is not clear, at this stage, whether it is at the level of the lexicon and the POS-tagger, with a piece of information going missing about substantive adjectives, or whether it is at the level of the grammar, which should better discriminate the two situations. We are tempted to believe that it is a combination of both.

The case illustrated in 4.4 probably also reveals a limitation in the way the corpus was artificially constructed. It could probably be argued[21] that the easiest correction to be consider would be a missing clitic between position 2 and 3—in English, it would be equivalent to a missing *one* after *fast*. Under such a conjecture, the parse

[20] It was pointed out to us by Eric Villemonte de la Clergerie (examiner, *examinateur*) that an even finer-grained distinction should be made in cases such as illustrated by the adverb *très*, which can modify both an adjective and an adverb, but can not stand alone.

[21] Thanks to Eric Villemonte de la Clergerie (examiner, *examinateur*) for bringing this point to our attention.

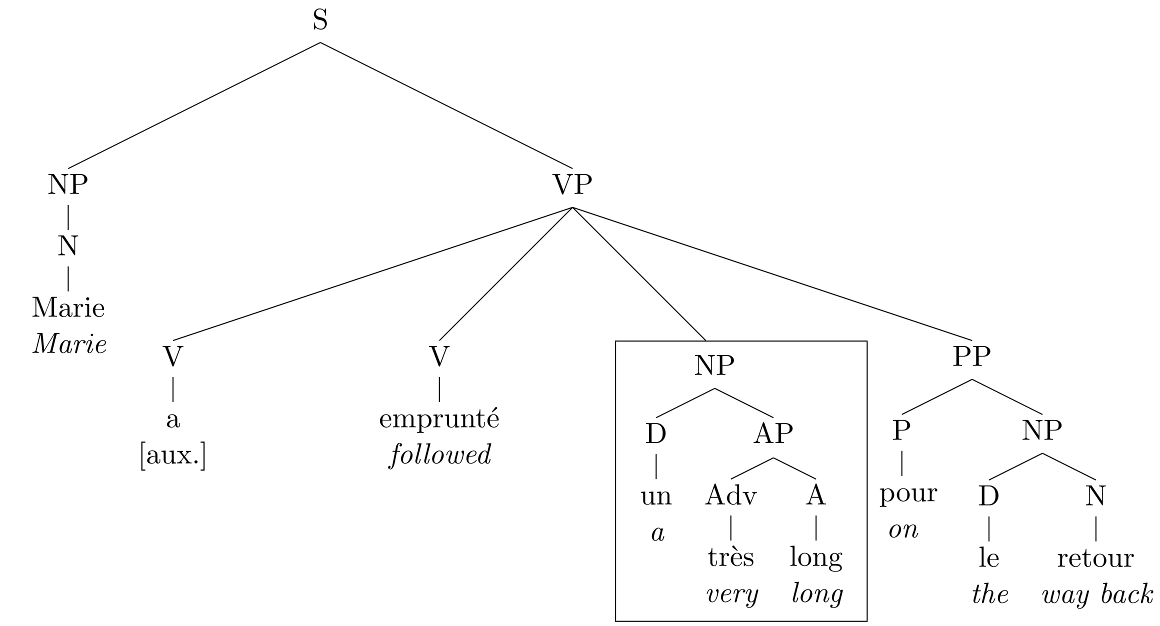


Figure 4.3: *Numbat*'s output for case of false substantive adjective—*Marie* [aux.] *followed a very long on the way back.*

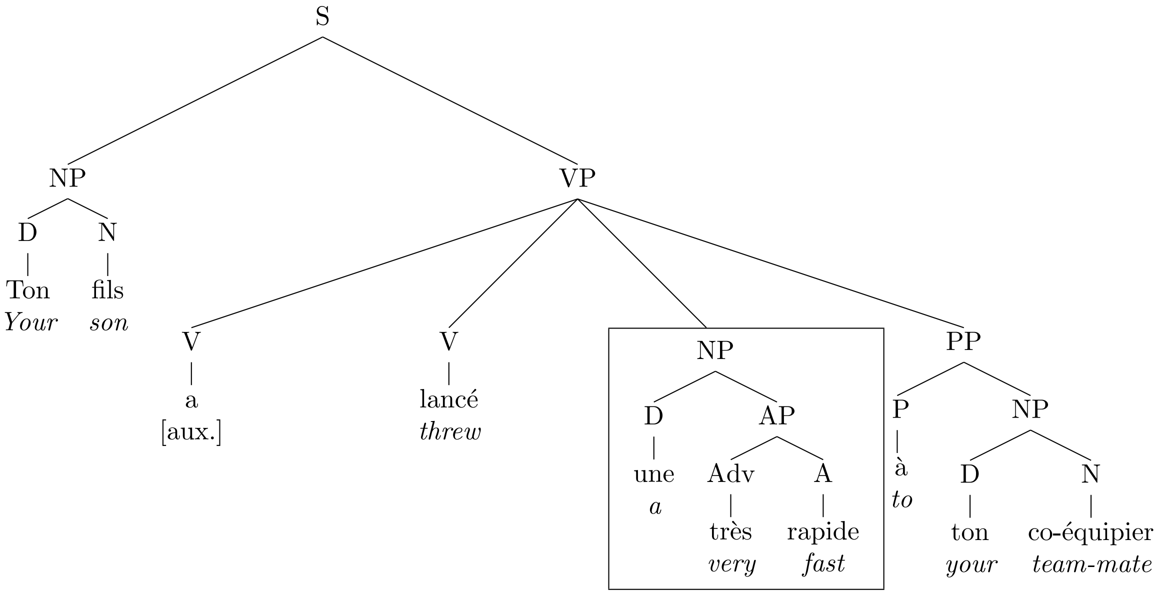


Figure 4.4: *Numbat*'s output for case of false substantive adjective—*Your son* [aux.] *threw a very fast to your team-mate.*

from Figure 4.4 would then be correct. This means that the base sentence (*i.e.* before application of the error pattern), in some cases, in not necessarily the best possible baseline for discussing the accuracy of the parser's solution.

Case 3: Multiple Past Participles The constituent structure in 4.5 illustrates a case where the redundancy of a past participle is not analysed by *Numbat* as deviant. It is a case of under-specification of the grammar, which licences multiple past participles in a VP. Cases of double past participle are indeed found, for instance, in French conjugation, or in passive voice constructions. Discriminating and specifying correctly such cases in the grammar is not straightforward and is out of the scope of this study.

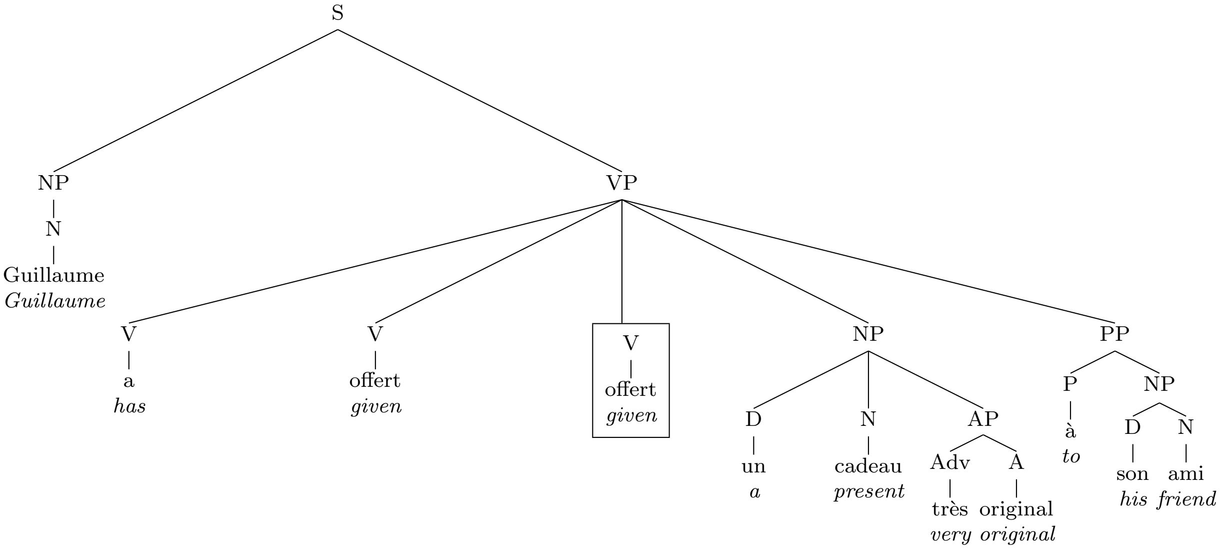


Figure 4.5: *Numbat*'s output for case of redundant past participle—*Guillaume has given given a very original present to his friend.*

Again, this case is a source of problems because it is rightfully structured by *Numbat*, but for the wrong reasons. The redundant past participle is expected not to prevent identifying the VP, and indeed it does not. However, as in the case of an AP heading an NP, no constraints are found to fail, which is expected to entail a wrong assessment of the utterance's gradience. The parser fails to find the case deviant, although it is actually so.

Borderline Case: Forest of Partial Parses

Since an important goal of the parser was to find a full parse for every sentence, the forests of partial parses were evaluated to *incorrect*. *Numbat* performs significantly well in this respect, since 92% of the sentences in the corpus get a full spanning constituent structure, and all the well-formed sentences get a full parse. However, and since forests are also taken into account by our experimental models of gradience, we now look at these into more details.

Why do we end up with partial parses, and in which cases? The cases concerned with partial parses may originate from different sources:

1. even in seeing language from the broad descriptive perspective we have been taking, it is not always possible to envisage every uttered language as a sequence of higher-lever constructions; many real-life contexts can be found where uttered language is incomplete. In §2.3.1 we have already discussed the classification of language phenomena suggested by Pullum and Scholz (2001), which gives us clues in this regard. Spoken language, for instance, is an actual source of such situations: interruptions during conversations, fluent speech, hesitations, and so on. Such a class of utterances is referred to as *structural fragments* by Pullum and Scholz (2001). Because of their fragmented nature it is quite predictable that these utterances will not be easily parsed as complete utterances. A forest of partial parses seems more likely and more appropriate.

2. The use of a numeric heuristic means that finding a full parse can not be guaranteed in every case.

Even assuming that all sentences in our corpus should get a full parse, for those cases which do not, deciding whether the problem comes from the grammar or from the heuristic in the parser is not straightforward. There are two possible elements of answer to that problem:

1. a cohesion threshold fixed too high, which prevents the parser from finding a full spanning parse; and

2. the grammar.

We observe that most cases of forest, that is, 93% of the forests (*i.e.* 42 out 45 cases) are cases of VP-violation (Type 5) (see §5.2 for more details about types of sentences), and 69% (31 out of 45) are cases of sentences with a missing head verb, as in Figures 4.6 and 4.7. A missing verb prevents finding a VP, which in turn prevents finding a main sentence—the wildcard construction (*star) is used as root, instead of the expected category utterance (S). We also observe that out of the 39 constraints specifying the VP construction in the grammar, 30 of them involve a verb constituent, while only 9 of them concern constituents other than the verb. That is 77% of the VP specification which concerns the head verb. Moreover, 8 of the 9 constraints without a verb concern the pronoun (P) construction, the ninth one specifying the uniqueness of an NP. It is important to emphasise that the constraints over pronoun constructions only cover special cases, and a VP without any P is perfectly fine. Therefore, it becomes extremely difficult—in fact, impossible in that case—to identify a VP with no P and a missing a head verb, since 38 of the 39 constraints can not be evaluated.

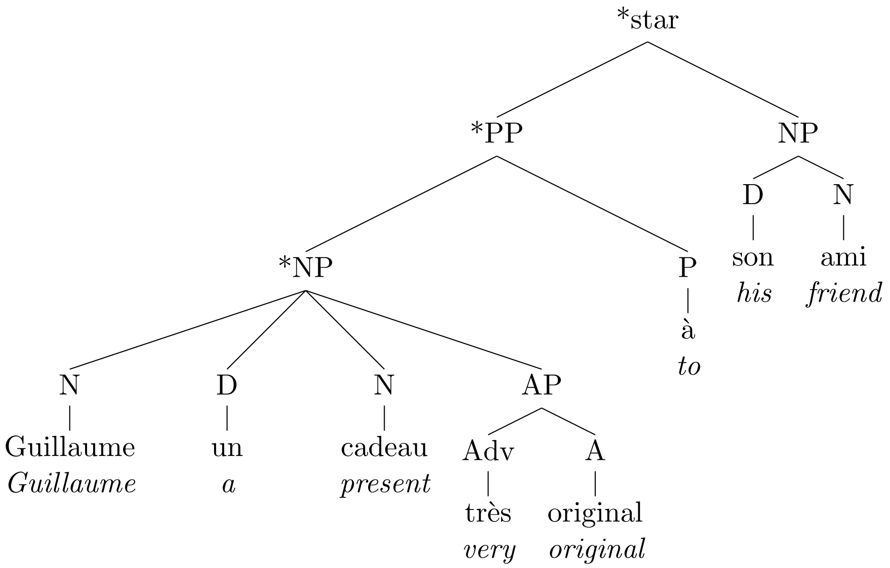


Figure 4.6: *Numbat*'s output for case of missing Verb: forest of partial parses—*Guillaume a very original present to his friend.*

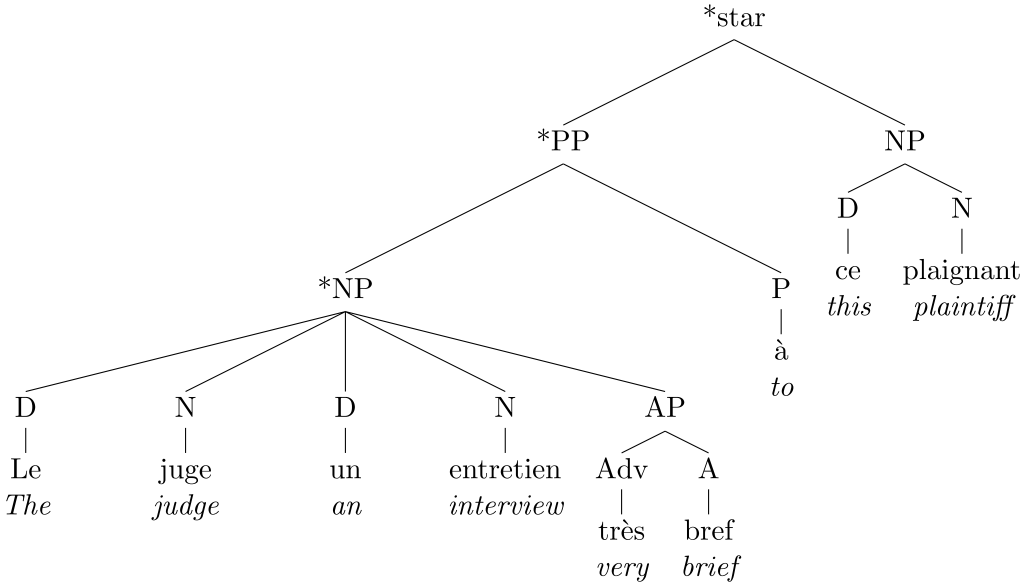


Figure 4.7: *Numbat*'s output for case of missing Verb: forest of partial parses—*The judge a very brief interview to this plaintiff.*

Thus, in conclusion, even a low threshold for the VP construction is not enough to enable the parser to consider as legitimate candidates for the Utterance construction constituent structures which would be missing a VP. Such cases require to be specified in the grammar in order to be handled by the parser.

4.9.5 Conclusion

In conclusion, the *Numbat* parser achieves good precision and recall of respectively 0.74 and 0.68, which are taken for measures of its ability to provide a correct full syntactic structure for deviant language. For lack of accuracy measures, different phenomena were observed and discussed. Cases of constructions being correctly categorised but wrongly characterised reveal under-specifications at the levels of the lexicon, the POS-tagger, and the grammar. These cases might turn out to be troublesome when assessing their score of gradience. Forests of partial parses represent only 8% of the corpus, which means a good achievement of the main goal for *Numbat*, concerned with providing a full spanning tree for deviant language. Closer investigation of these forests shows that they also reveal under-specification in the grammar for cases of structural fragments.

4.10 Conclusion

Our goal in this chapter was to develop a robust parser which let us experiment with syntactic gradience. We have presented an algorithm for Loose Satisfaction Chart Parsing (LSCP) in order to meet that goal. LSCP is a dynamic programming approach, which generates a full parse structure of optimal merit for any input utterance. The algorithm shows a worst time complexity of $\mathcal{O}(n^4 2^n C^n)$. Numerical heuristics based on the merit of constituents is used to prune the search space and drop the time complexity.

The parser was run over a corpus made of 94% of deviant utterances. The output was evaluated by human annotators, in order to determine whether the parse trees provided by LSCP for deviant language were acceptable. It results a measure precision of 0.74 and a measure of recall of 0.68. These measures are very good ones, especially when put in the perspective of how much of the corpus involves deviant language.

Chapter 5

A Computational Model For Syntactic Gradience

5.1 Introduction

So-called *measures of grammaticality* are used in many areas of Computational Linguistics such as Summarisation, Machine Translation, Question-Answering or Natural Language Generation, but are rarely substantiated by a proven ability to capture human judgement of acceptability. In fact, very few studies from computational linguistics have attempted to systematically investigate the possibility to capture a gradient of acceptability with a computational model, and evaluate to what extent model and human judgement correlate. The venture is indeed quite challenging. The observation of the phenomenon to be modelled is arguable in nature since it is concerned with human judgement, and therefore evaluating the the fit of a model is open to argument. This work aims to model the propensity of language to possess varying degrees of acceptability—referred to as *gradience*.

In line with previous work (Aarts, 2004b, 2007; Blache and Prost, 2005; Blache, Hemforth, and Rauzy, 2006; Foth, 2004; Keller, 2000; Schröder, 2002; Sorace and Keller, 2005) we focus on syntactic gradience. We suggest that the concepts of *Intersective Gradience* and *Subsective Gradience* introduced by Aarts for modelling graded judgements be extended to deviant language. Under such a new model the problem then raised by gradience is to classify an utterance as a member of a specific construction according to its syntactic characteristics. More specifically, we propose to extend Intersective Gradience (IG) so that it is concerned with choosing the most suitable syntactic structure for an utterance among a set of candidates, while Subsective Gradience (SG) is extended to be concerned with calculating to what extent the chosen syntactic structure is typical from the construction at stake. We claim that the two problems, despite their overlap, should be addressed separately. We propose that IG be addressed in relying on a criterion of optimality, while SG should be addressed in rating an utterance according to its grammatical acceptability.

The question of the optimality of a syntactic structure was addressed in Chapter 4, using Property Grammars (PG) and the characterisation of an input to model its syntactic properties. In this chapter, we focus on completing our model of gradience with an address of SG, where an utterance is rated according to its syntactic gradience. We start by postulating factors of influence on SG, then we propose two different numeric models to capture these factors. We then investigate to what extent these numerical accounts of gradience correlate to human judgements of grammatical acceptability. These human judgements were gathered independently from the present work, as part of a psycholinguistic experiment reported in Blache, Hemforth, and Rauzy (2006). Their experiment involve 44 annotators, all native speakers of French, and with no particular knowledge of linguistics[1]. We show that despite language deviance and despite imperfections in the automatically generated syntactic parses, our model still provides very good results at predicting an utterance's acceptability. It even outperforms the results of a similar experiment (Blache, Hemforth, and Rauzy, 2006, henceforth BHR06) where manual parses were used. Note that unlike all the approaches mentioned above, the model we propose for rating an utterance relies exclusively on the automatically generated output of our parser.

As a preamble, in §5.2 we present the reference corpus we are using for our investigation and the way this corpus,along with the human judgements, was gathered. In

[1] Note that although the underlying corpus in use is the same the experiment from Blache, Hemforth, and Rauzy is not to be confused with the evaluation of *Numbat* that we have run and presented in 4.9.3, which involves 5 annotators, all linguists, but who were not asked to provide any sort of numerical values.

§5.3 we present and discuss the approach we have adopted for modelling syntactic gradience, then we postulate five factors of influence. In §5.4 we present the two models we have designed for rating an utterance, as well as the model introduced in BHR06. In §5.5 we present the comparative empirical investigation of the two models we have introduced, along with the one from Blache *et al.* And finally, we draw conclusions in §5.6.

5.2 Reference Corpus

The corpus and data we are using as a reference for human judgement of acceptability was gathered independently from the present study; corpus and experimental protocol are described in Blache, Hemforth, and Rauzy (2006, p. 62) as follows.

> We ran a questionnaire study presenting participants with 60 experimental sentences (...). 44 native speakers of French completed the questionnaire giving acceptability judgements following the Magnitude Estimation technique. 20 counterbalanced forms of the questionnaire were constructed. Three of the 60 experimental sentences appeared in each version in each form of the questionnaire, and across the 20 forms, each experimental sentence appeared once in each condition. Each sentence was followed by a question concerning its acceptability.
>
> These 60 sentences were combined with 36 sentences of various forms varying in complexity (simple main clauses, simple embeddings and doubly nested embeddings) and plausibility (from fully plausible to fairly implausible according to the intuitions of the experimenters). One randomisation was made of each form.
>
> *Procedure*: The rating technique used was magnitude estimation (ME, see Bard, Robertson, and Sorace, 1996). Participants were instructed to provide a numeric score that indicates how much better (or worse) the current sentence was compared to a given reference sentence (Example: If the reference sentence was given the reference score of 100, judging a target sentence five times better would result in 500, judging it five times worse in 20). Judging the acceptability ratio of a sentence in this way results in a scale which is open-ended on both sides. It has been demonstrated that ME is therefore more sensitive than fixed rating-scales, especially for scores that would approach the ends of such rating scales (cf. Bard, Robertson, and Sorace, 1996). Each questionnaire began with a written instruction

> where the subject was made familiar with the task based on two examples. After that subjects were presented with a reference sentence for which they had to provide a reference score. All following sentences had to be judged in relation to the reference sentence. Individual judgements were logarithmized (to arrive at a linear scale) and normed (z-standardized) before statistical analyses.

The resulting figures are reported in Table 5.1. Four of the error types correspond to semantic errors (pointed with ‹ in Table 5.1) and are thus removed from the corpus, because out of the scope of our investigation. We are then left with 16 different types of sentences, one of which being the base sentence (well-formed), and 15 others corresponding to deviant utterances. For each type the mean acceptability score is calculated.

Next to the data collection BHR06 reports an experiment, where these scores of acceptability judgement were correlated to the scores calculated automatically. However, the characterisations and the parse trees used as input material for calculating the scores with the model were produced manually. Our own experiment relies on output automatically generated and provided by the parser *Numbat*that we have developed, and presented in Chapter 4.

5.3 Modelling Syntactic Gradience

Ultimately, we are interested in designing a model of prediction for acceptability judgement based on syntactic gradience. The approach we adopt for modelling syntactic gradience proceeds in two steps:

1. categorisation of the utterance as a construction[2], based on syntactic criteria;

2. assessment of a score of syntactic gradience for the utterance, based on the characterisation[3] of the utterance.

[2] For what we are interested in in this chapter, a construction is a class of utterances, specified by a common set of conditions of well-formedness. See 3.2.4 for more details.

[3] The characterisation of an utterance is the set of constraints that the utterance meets and/or violate. See Definition 3.14 for more details. For the purpose of modelling gradience, it can be seen as the syntactic characteristics (in the sense of Aarts), which serve as features for classifying an utterance in a construction.

No violations		
1.1	Marie a emprunté un très long chemin pour le retour *Marie* [aux.] *followed a very long path on the way back*	0.465
NP-violations		
2.1	Marie a emprunté très long chemin un pour le retour *Marie* [aux.] *followed very long path a on the way back*	-0.643
2.2	Marie a emprunté un très long chemin chemin pour le retour *Marie* [aux.] *followed a very long path path on the way back*	-0.161
2.3	Marie a emprunté un très long pour le retour *Marie* [aux.] *followed a very long on the way back*	-0.871
2.4	Marie a emprunté très long chemin pour le retour *Marie* [aux.] *followed very long path on the way back*	-0.028
2.5	Marie a emprunté un très heureux chemin pour le retour *Marie* [aux.] *followed a very happy path on the way back*	-0.196 <
AP-violations		
3.1	Marie a emprunté un long très chemin pour le retour *Marie* [aux.] *followed a long very path on the way back*	-0.41
3.2	Marie a emprunté un très long long chemin pour le retour *Marie* [aux.] *followed a very long long path on the way back*	-0.216
3.3	Marie a emprunté un très chemin pour le retour *Marie* [aux.] *followed a very path on the way back*	-0.619
3.4	Marie a emprunté un grossièrement long chemin pour le retour *Marie* [aux.] *followed a roughly/rudely long path on the way back*	-0.058 <
PP-violations		
4.1	Marie a emprunté un très long chemin le retour pour *Marie* [aux.] *followed a very long path the way back on*	-0.581
4.2	Marie a emprunté un très long chemin pour pour le retour *Marie* [aux.] *followed a very long path on on the way back*	-0.078
4.3	Marie a emprunté un très long chemin le retour *Marie* [aux.] *followed a very long path the way back*	-0.213
4.4	Marie a emprunté un très long chemin pour *Marie* [aux.] *followed a very long path on*	-0.385
4.5	Marie a emprunté un très long chemin dans le retour *Marie* [aux.] *followed a very long path in the way back*	-0.415 <
VP-violations		
5.1	Marie un très long chemin a emprunté pour le retour *Marie a very long path* [aux.] *followed on the way back*	-0.56
5.2	Marie a emprunté emprunté un très long chemin pour le retour *Marie* [aux.] *followed followed a very long path on the way back*	-0.194
5.3	Marie un très long chemin pour le retour *Marie a very long path on the way back*	-0.905
5.4	Marie emprunté un très long chemin pour le retour *Marie followed a very long path on the way back*	-0.322
5.5	Marie a persuadé un très long chemin pour le retour *Marie* [aux.] *convinced a very long path on the way back*	-0.394 <

Table 5.1: Error patterns. *The left-most column contains a numbered label for the pattern type and sub-type; the right-most column gives the mean score of acceptability (human judgement); sentences marked < are removed from the corpus.*

Each of the two steps involves a decision made on the basis of a numerical value for a constituent. We argue that because these decisions are different in nature, they rely on different ranking schemes and therefore require different scoring functions.

The first step is concerned with deciding which construction an item belongs to among several candidates, and was already addressed in 4.4.4. The decision is made by ranking the characterisation of the item with respect to each construction. Note that, since a construction presents a hierarchical structure of construction elements, this categorisation problem is recursive in nature. The *Numbat* parser presented in Chapter 4 provides us with an optimal constituent structure for any input string. By default, should a full parse not being found, the output is a sequence of disconnected constituents (*i.e.* a forest of partial parses) presented as single tree rooted by the wildcard construction *Star* (see the Memoization Step in §4.4.3 for more details on how this default solution is built). The parser's output is interpreted as providing a class (*i.e.* a construction) for the input, along with a characterisation of the constituent in terms of satisfied and violated constraints.

The second step is concerned with rating an utterance, so that the score correlates with its acceptability. Note that in the remainder of this chapter we use the term *acceptability* to refer to *human judgement of grammatical acceptability*, unless mentioned otherwise. The correlation presumes that a score signifying acceptability is available for the utterance. The rating problem as such consists of assessing how acceptable each utterance is with respect to its construction. Thus, the problem is to calculate a score for the input, based on the characterisation, which reflects the utterance's syntactic gradience. The comparison we run later in this chapter among rating models for syntactic gradience is performed in reference to the results gathered by the empirical study on human judgement of acceptability run independently from our own investigation and presented in BHR06.

The first step of this approach to language processing is quite similar in nature to the one formalised in Optimality Theory (OT) (Prince and Smolensky, 1993) and extended by Keller (2000) (LOT, see §2.4.2) in order to rank all candidate structures—though there exists differences regarding the criteria in use in the decision process regarding optimality. The second step, however, differs significantly from OT or even

Keller's LOT which are not open to rating an utterance[4]. We come back to this argument in §5.3.2.

What we are trying to suggest here is that the Intersective/Subsective (IG/SG) interpretation of syntactic gradience by Aarts (2004b, 2007) (see §2.2) can be extended in order to cover deviant language. Such an extension consists of seeing the candidate structures for an utterance from the first step as as many constructional categories. Finding out which candidate is the most suitable one—according to a criterion of optimality or other—is then a problem of intersective gradience. Subsective gradience, on the other hand, involves deciding to what extent the optimal structure is typical for the constructional category concerned. Thus, it corresponds to the second step we described.

Next, we introduce and discuss how we address each of the two steps mentioned above.

5.3.1 Merit *vs.* Rate

Is it the same problem on one hand to decide, among a list of candidates, which construction the constituent belongs to, and on the other hand to calculate the degree of acceptability of a constituent? We claim it is not. The two problems are significantly different, although obviously not completely separate. The first proble problem, discussed in §4.5, relies on a merit function in order to determine which of the candidate syntactic structures is of optimal merit, while the second problem is concerned with rating an input sentence according to its syntactic gradience. Put in slightly different words, the merit function is used by the parser for determining the optimality of a candidate structure, while a model of gradience is to be used for rating an utterance. Although it could intuitively be argued that the same numerical model could be used for achieving both tasks, we are going to argue that not only the two problems are sufficiently distinct to be handled separetly, but most importantly that using one and

[4] This point was already discussed in the sub-section about Grammaticality in §2.3.2, as well as in §2.4.2, especially in the sub-section on Shortcomings of LOT. As far as rating an utterance is concerned, the problem comes from that these theories can not compare the respective degrees of grammaticality of two quasi-expressions.

same numerical model for both the merit function and the model of gradience raises important concerns.

We are arguing that a model of gradience should be independent from the input characterisation, and thus from the input syntactic parse. Since the merit function is directly involved in generating the characterisation of a sentence, it is not a suitable candidate for modelling gradience. Therefore, merit and rate require to rely on numerical functions which are independent from each other. In being independent from the input parse the model of gradience can then be generalised and used in combination with other parsers, as long as a characterisation is provided for the sentence to be rated. Besides, using the merit function as the model of gradience would also present several drawbacks, which we would like to discuss now.

One drawback stands in that any modifications in the model calibration would necessarily change, at the same time, the criteria of optimality which the parser relies on in order to generate a parse. This is not something for the model of gradience to change, as it would be a very different way of looking at gradience. From our point of view, the syntactic characterisation of a sentence must be taken as an observation, to which the model of gradience is applied in order to rate the sentence.
Another drawback is methodological, since the comparison among candidate models of gradience would be highly arguable if the models under investigation do not take same input. This would indeed be the case if the models are also used by the parser as the merit function which determines the optimal parse. For different merit functions one could expect different optimal parses to be generated. Therefore, two or more such functions, when used as models of gradience, would rely on different characterisations of the same sentence. Such a situation would jeopardise their comparison since it could not be performed indenpendently from external variable factors.

Despite that argument, we have considered the Cohesion of a constituent as one of the models in competition, in order to back up our claim and make sure that it is not suitable to modelling gradience. The scores obtained are reported in Table 5.2, while the scatter plot in Figure 5.1 shows the linear regression between Cohesion scores and acceptability judgements. As explained earlier in §5.2 into more details, the *type* corresponds to an error pattern, and the reference scores of Acceptability are those

Type	Acceptability	Cohesion
1.1	0.4647	0.9981
2.1	-0.6484	0.9513
2.2	-0.1614	0.9738
2.3	-0.8711	0.9994
2.4	-0.0281	0.9736
3.1	-0.4067	0.9930
3.2	-0.2157	0.9982
3.3	-0.6193	0.9926
4.1	-0.5811	0.9761
4.2	-0.0779	0.9783
4.3	-0.2129	0.9342
4.4	-0.3852	0.9730
5.1	-0.5603	0.9762
5.2	-0.1935	0.9967
5.3	-0.9054	0.9342
5.4	-0.3217	0.9685
$\rho = 0.2814$		

Table 5.2: Human judgements of acceptability and Cohesion, per sentence type (*i.e.* error pattern)

of human judgement. What we are interested in here is that the Cohesion performs rather badly as a model of gradience, since it only shows a very low correlation ($\rho = 0.2814$) to the Acceptability ratings of the human judges.

Note as well that the Cohesion scoring function does not require any calibration since it does not involve any parameters.

5.3.2 Postulates

We postulate that acceptability can be predicted by factors derivable from the output of Loose Satisfaction Chart Parsing (LSCP) (see Chapter 4). Three of our postulates (1, 2, and 3) are substantiated by empirical evidence and work in the field of Linguistics and Psycholinguistics, while two others (4 and 5) are hypothetical.

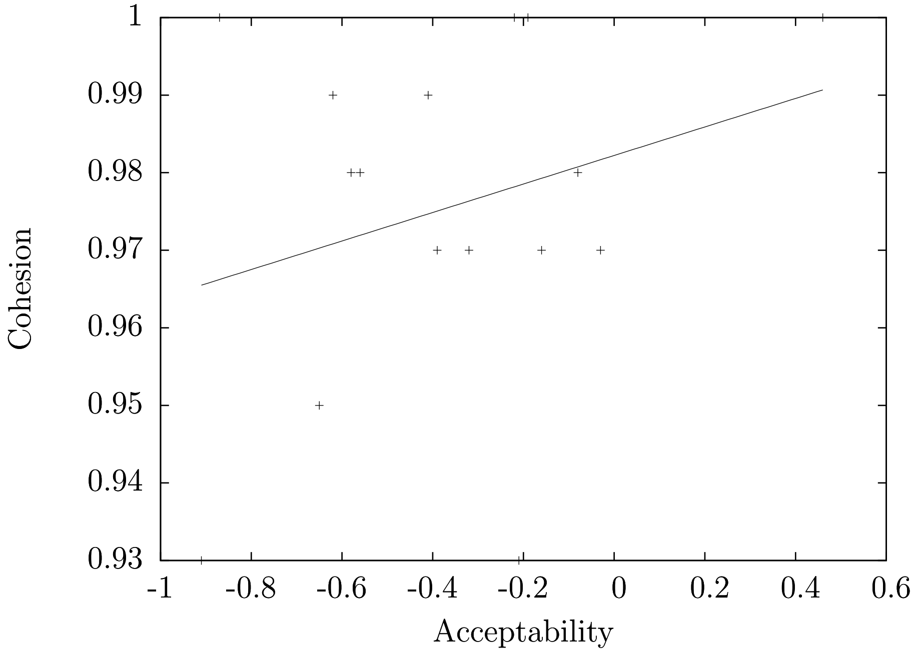


Figure 5.1: Cohesion *vs.* Acceptability: model fit over the full corpus

Failure Cumulativity

Since we are interested in deviant language, constraint failures must be accounted for.

Postulate 1 (Failure Cumulativity) *Gradience is impacted by constraint failures; that is, an utterance's acceptability is impacted by the number of constraints it violates.*

We denote by N_c^- the number of constraints violated by the constituent c.
This factor corresponds to Keller's cumulativity effect, and is substantiated by empirical evidence:

> Constraint violations are cumulative, *i.e.*, the unacceptability of a structure increases with the number of constraints it violates. (...) this is an effect of considerable robustness and generality; it applies to both soft and hard violations, and to multiple violations of the same constraint and of different constraints.(Keller, 2000, p. 122)

Success Cumulativity

Meanwhile, it was suggested in different works (Aarts, 2004b; Blache and Prost, 2005; Blache, Hemforth, and Rauzy, 2006; Aarts, 2007) (see §2.4.1, especially §2.4.1), that gradience is also affected by successful constraints.

Postulate 2 (Success Cumulativity) *Gradience is impacted by constraint successes; that is, an utterance acceptability is impacted by the number of constraints it satisfies.*

We denote by N_c^+ the number of constraints satisfied by the constituent c, and $E_c = N_c^+ + N_c^-$.

The underlying intuition is that failures alone are not sufficient to account for acceptability. Unlike in other accounts of syntactic gradience which only or mostly rely on constraint failures (*e.g.* Keller's LOT §2.4.2, Schröder's WCDG §2.5.2) we postulate that some form of interaction between satisfied and violated constraints contributes to the gradience of acceptability. Keller's model is prevented from even being open to the possibility of taking any form of success cumulativity into account due to be based on OT. Since OT, by conception, discriminates candidate structures only on the basis of the constraints they violate, relying on satisfied constraints is not an option.

Constraint Weighting

Following the intuition commonly shared by everyone interested in modelling gradience, and supported by empirical evidence (Blache and Prost, 2005; Blache, Hemforth, and Rauzy, 2006; Bresnan and Nikitina, 2003; Foth, 2004; Gibson, 2000; Heinecke et al., 1998; Keller, 2000; Sorace and Keller, 2005; VanRullen, 2005), all constraints do not have the same importance with respect to acceptability:

Postulate 3 (Constraint Ranking) *Acceptability is impacted to a different extent according to which constraint is satisfied or violated.*

Here we postulate that constraints are weighted according to their influence on acceptability. The constraint weights are as many variable parameters for tuning up the model, and their values are estimated during the calibration phase. The question of whether such weights are proportional to the importance of either constraint success or failure is addressed in assuming that a given constraint is of same importance either way in absolute value. We denote by W_c^+ (respectively W_c^-) the sum of the weights assigned to the constraints satisfied (respectively violated) by the constituent c.

Weighting Scope and Granularity *Scope* and *granularity* of a constraint weight are actually tightly connected to each other. The *scope* has to do with how widely a weight applies: does the same weight apply to all constraints of one constraint type[5], or to each individual constraint? Whereas the *granularity* has to do with the level at which a weight applies: does the same weight apply at the grammar level, or at the construction level? Scope and granularity can then be grouped in different ways: all constraints of the same type at the grammar level, or all constraints of the same type at the construction level, or individual constraints at the construction level, or individual constraints at the grammar level—the difference between the last two possibilities assumes that the same constraint may occur in the specification of more than one construction.

Although the influence on gradience gets more flexible and accurate with a more fine-grained and narrowe scope, a too fine granularity and a too narrow scope would also be quite complex to manage and require thorough empirical studies beyond the scope of this work. Therefore, we opt for a compromise, where the weighting scheme is restricted to the constraint types at the grammar level, which means that all constraints of the same type in the grammar are assigned the same weight. For examples, all constraints of linearity (*i.e.* word order) are weighted 20, all constraints of obligation (*i.e.* heads) are weighted 10, and so on. denote by that in WCDG (see §2.5.2), for instance, Schröder opted for more flexibility with weights being assigned to each individual constraint at the grammar level. However, as expected, the task of assigning values to every weight is proven very costly.

Automatic Weighting Unfortunately, as emphasised by Aarts (see §2.4.1), very few uncontroversial automated methods exists for the acquisition of weight values.

[5] See §3.2.3 for definitions of constraint types in PG.

In Keller's Linear Optimality Theory (see §2.4.2) the constraint weights, along with empirical data gathered from corpora, form an equation system, which is solved using the least squares method. Although the method is not directly applicable as such in our case, it seems to be possible to adapt a corpus-based approach using machine learning. The question must remain open. As far as this study is concerned, wet calibrated the weights along with the other parameters using the method of *gradient ascent*. We describe it in §5.5.

In VanRullen (see §2.5.4) constraint weights do not play any significant practical role, though the framework is open to their implementation.

Constraint Ranking Subsequent to the weighting scheme, a ranking is applied to the constraints, which conforms to Keller's findings in this respect. As we will see in §5.5, we assume no particular pre-ranking. Since the constraints are ranked according to their weights, and since the weights serve as parameters for tuning up the model, the constraints are automatically ranked by side-effect. Thus the constraint ranking is observed empirically.

Similarly, we do not pre-establish a dichotomy into *soft* and *hard* constraints as suggested by Keller (see §2.4.2), but observed it empirically, with weights tending to extremal values (the constraints of linearity and obligation being clearly hard ones).

Constructional Complexity

Postulate 4 (Constructional complexity) *Acceptability is impacted by the* complexity *of the constituent structure.*

How to precisely measure and model the complexity of a constituent is an open question, which we do not claim to fully answer. In fact, this factor of influence probably ought to be investigated in itself, and split into more fine-grained postulates with respect to acceptability and syntactic gradience. Yet, we do make a number of attempts in this work to capture factors which are likely to be involved in a constituent's complexity. One of these attempts is to measure the complexity of the construction a constituent belongs to as the amount of constraints specifying this construction in the grammar.

We denote by $T_{\mathcal{C}}$ the total number of constraints specifying the construction $\mathcal{C}$ of the constituent c. $T_{\mathcal{C}}$ is not to be confused with E_c. $T_{\mathcal{C}}$ is a constant specific to a construction, while E_c depends on a constituent. They may relate in different ways:

- we may have $T_{\mathcal{C}} > E_c$ if there are constraints from $\mathcal{C}$ which can not be evaluated in c; or
- we may have $T_{\mathcal{C}} < E_c$ when multiple instances of the same constraint from $\mathcal{C}$ can be evaluated in c.

This postulate aims to address, among others, the problems concerned with the risk of disproportionate convergence raised in §2.4.1. The underlying idea is to balance the number of violations with the number of specified constraints: without such a precaution a violation in a rather simple construction, such as AP—specified by seven constraints in our toy grammar Γ presented in §3.2.4,would be proportionally much more costly than a violation in a rather complex construction, such as NP—specified by fourteen constraints in Γ[6].

Propagation

Postulate 5 (Propagation) *Acceptability is propagated through the relationships of dominance; that is, an utterance acceptability depends on its nested constituents' acceptability.*

The number of nested constituents is denoted by Z_c. Here we postulate that the nested constituents' acceptability is propagated to their dominant constituent. Although this may seem trivial, it is not necessarily so. The process of loose satisfaction yields a full and legal constituent structure, where deviant constituents are visible from their immediate dominant constituents, but all violations are overcome beyond them. Therefore, the information of a deviant constituent may be lost, unless it is propagated explicitly. The situation is exemplified with a simple case in Figure 5.2 (see Table 3.1 for a legend). Subsequently the models we investigate are dependent functions of their sub-constituents' score.

[6] See Example 4.9 for an illustration of that case.

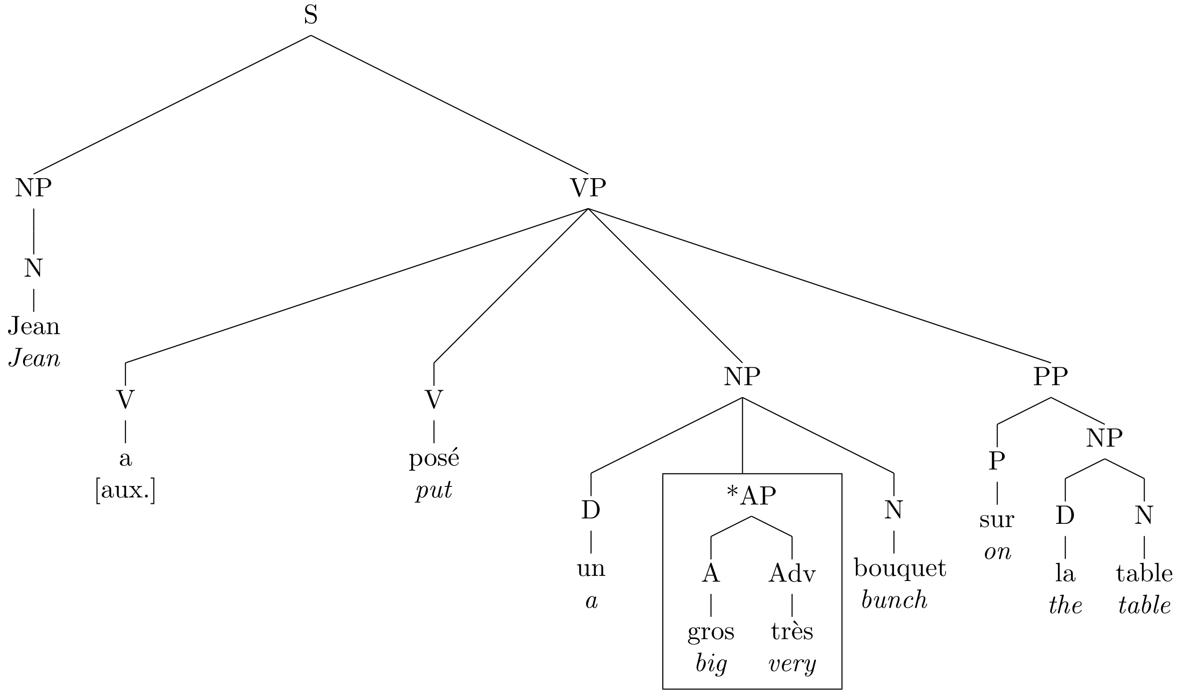


Figure 5.2: Propagation Postulate. *The violation of word order between Adverb and Adjective is visible from the immediately dominant Adjective Phrase, but may be lost if not propagated to the Noun Phrase and further.*

5.4 Rating Models

A rating model for gradience aims to place an item along a scale by assigning a score (or rate) to it. That score is representative of both the item's absolute worth, and what it is worth with regard to other items—*i.e.* its position on a gradient.

First we formulate a series of intermediate scoring functions as an attempt to capture these factors. Then we introduce the full models under investigation.

5.4.1 Scoring Terms

In this section we introduce various scoring components, which aim to capture the postulates previously mentioned. These components are involved in the different models we investigate later. Each component is meaningful as such, but not sufficient when considered alone.

Blache and Prost (2005) first introduced the notion of *density*, later re-defined in BHR06 as *Satisfaction Ratio*. In order to avoid confusions with the *density of satisfaction* from VanRullen (2005), for which a different definition is provided (see §4.4.4 for more details), we adopt the terminology and the definition of a Satisfaction Ratio from BHR06. The Satisfaction Ratio and Violation Ratio capture postulates 2 and 1, respectively. Their definitions are reproduced in Definition 5.1 and 5.2 respectively.

Definition 5.1 (Satisfaction Ratio) *We define the* Satisfaction Ratio *(SRatio) for the constituent c as follows:*

$$\varrho_c^+ = \frac{N_c^+}{E_c}$$

Definition 5.2 (Violation Ratio) *We define the* Violation Ratio *(VRatio) for the constituent c as follows:*

$$\varrho_c^- = \frac{N_c^-}{E_c}$$

The ability of these two scores to model gradience is investigated in Blache and Prost (2005) for various constructions on relatively small corpora (*Le Monde*, 15,420 words, transcriptions of spoken corpora, 523 and 1,923 words). The emphasis is put on deviant language. The SRatio is meant to capture the *cumulativity* effect discussed in Keller (2000) and Sorace and Keller (2005) (see §2.4.2). However, shortcomings are also observed. Blache and Prost (2005) report, for instance, that the SRatio does not necessarily grow with grammaticality; cases are reported of grammatical constructions for which low SRatio is observed, whereas a high one, proportional to grammaticality, was expected. It is suggested by the authors that such a discrepancy may be partly explained by the influence of differences in the construction specifications in terms of number of constraints involved. The conclusion we can draw here is that although SRatio and VRatio are meaningful factors for quantifying aspects of grammaticality, their prediction power is not strong enough to accurately model acceptability judgement.

With respect to our own postulates (see §5.3.2), SRatio is meant to implement the *success cumulativity* factor, while VRatio is meant to implement the *failure cumulativity* factor. The suggestion in Blache and Prost (2005) regarding the influence of the construction specifications is also captured by Postulate 4.

BHR06 introduces more scoring components, in order to account for different aspects of gradience. We reproduce them here[7].

Definition 5.3 (Completeness Index) *We denote by $T_{\mathcal{C}}$ the total number of constraints specifying construction $\mathcal{C}$; the* Index of Completeness *for the constituent c of construction $\mathcal{C}$ is define in BHR06d as the following ratio:*

$$\mathcal{E}_c = \frac{E_c}{T_{\mathcal{C}}}$$

The *Completeness* of a constituent puts the number of constraints it is concerned with into the perspective of the number of constraints specifying the construction it belongs to. This score contributes to the implementation of Postulate 4 (see §5.3.2), which suggests that the complexity of a constituent influences its acceptability.

[7] Part of the notation adopted in Blache, Hemforth, and Rauzy was changed, essentially because we find one-letter terms easier to read in mathematical formulae than two-letter terms: SR is now ϱ^+, CC is now $\mathcal{E}$, QI is now $\mathcal{W}$, PI is now $\mathcal{P}$, and GI is now g.

Definition 5.4 (Quality Index) *The* Index of Quality *for the constituent c is defined in BHR06 as the following ratio:*

$$\mathcal{W}_c = \frac{W_c^+ - W_c^-}{W_c^+ + W_c^-}$$

The *quality* of a constituent implements Postulate 3 (see §5.3.2), which suggests that all constraints do not have same importance with respect to acceptability, and therefore must be weighted accordingly. This score represents the overall importance of the constituent in the form of a mean weight. The Index of Quality is the average weight of all the constraints it is characterised with, where the weights of satisfied constraints are positive, and those of violated constraints are negative.

5.4.2 Combining Terms into Scoring Functions

Precision

Definition 5.5 (Precision Index) *The* Index of Precision *for the constituent c is defined in BHR06 as the following ratio:*

$$\mathcal{P}_c = \frac{k \cdot \mathcal{W}_c + l \cdot \varrho_c^+ + m \cdot \mathcal{E}_c}{3}$$

The *Precision* of a constituent combines its Quality ($\mathcal{W}_c$), its Satisfaction Ratio (ϱ_c^+), and its Completeness ($\mathcal{E}_c$) within one score. We assume that the three terms are of different importance; thus, each of them is associated with a multiplicative coefficient[8] in order to balance their respective strength. These coefficients (k, l, m) are used as variable parameters for tuning up the model. From now on we will refer to them as *adjustment coefficients.*

In addition to those terms defined in Blache and Prost (2005); Blache, Hemforth, and Rauzy (2006) and VanRullen (2005), and the Index of Precision introduced in BHR06, we define a number of new scoring functions, in order to overcome some of the shortcomings and drawbacks we observed in using the former.

[8] Note that the factor 1/3 could be removed and $\langle k, l, m \rangle$ set to a third of the chosen values. We keep the following definitions as such in reference to the original model defined by Blache *et al.*

Anti-Precision

We observe that the SRatio in use in the Precision score seems to over-emphasise the role of success cumulativity, that is, the role of the successful constraints characterising a constituent. Therefore, and in order to check to what extent it is the case, we define (Definition 5.6 below) an index of *Anti-Precision*, where the Satisfaction Ratio term in the Precision Index is replaced by the Violation Ratio as a negative term.

Definition 5.6 (Anti-Precision Index) *We define the* Index of Anti-Precision *for the constituent c as the following ratio:*

$$\widetilde{\mathcal{P}}_c = \frac{k \cdot \mathcal{W}_c - l \cdot \varrho_c^- + m \cdot \mathcal{E}_c}{3}$$

Different from the Precision score, the Anti-Precision emphasises the factor of failure cumulativity, while maintaining a tradeoff with success cumulativity.

Taxed Precision

Another observation we make is that the factor of complexity might not be captured accurately enough, leaving too large a gap between the importance of failures on poorly characterised constituents on one hand, and the importance of failures on richly characterised constituents on the other hand. We have already observed (see §2.4.1) that there are poor constructions and rich constructions in terms of how many constraints they require to be specified, and that a single failure has more important consequences on a poor constituent than on a rich constituent. From there, we hypothesise that out of fairness the rich constituents already violating constraints ought to be penalised proportionally to the number of constraints they are concerned with. We define the notion of *Taxed Precision* to capture that intuition.

Definition 5.7 (Taxed Precision) *Let* $\mathtt{T}$ *be a penalty, in number of constraints. We define the* Taxed Precision *of the constituent c as the following ratio:*

$$\widetilde{\mathcal{P}'}_c = \frac{k \cdot \mathcal{W}_c - l \cdot \frac{N^- + \mathtt{T}}{N_c^+ + N_c^-} + m \cdot \mathcal{E}_c}{3}$$

The penalty rates applied for T are the following:

$$\mathtt{T} = \begin{cases} 1 & \text{if } 5 < E \leqslant 10 \\ 2 & \text{if } 10 < E \leqslant 15 \\ 3 & \text{if } 15 < E \end{cases}$$

Note that these penalty rates are static, constant, and heuristically determined. In further works, it might be interesting to consider them as variable parameters, in order to bring more flexibility to the model.

5.4.3 Rating Functions

Among the numerous functions we have investigated, the following ones more particularly draw our attention for the significance of their results.

Grammaticality

Definition 5.8 (Grammaticality Index) *The* Index of Grammaticality *(g) for the constituent c is defined in BHR06 recursively as follows (where c_i is a nested constituent of c, and Z_c the number of nested constituents in c):*

$$g_c = \mathcal{P}_c \cdot \overline{g_{c_i}} = \mathcal{P}_c \cdot \frac{\sum_{i=1}^{Z_c} g_{c_i}}{Z_c}$$

BHR06 report experiments which aim to validate the ability of g to predict acceptability. A sample of 16 sentences from the annotated corpus we described in §5.2 was used for these experiments, and a very good correlation (coefficient $\rho_1 = 0.76$) is reported between g and human acceptability judgements. An even better correlation of coefficient $\rho_2 = 0.87$ is reported on a smaller sample of corrected data[9]. Such figures lead the authors to present the Index of Grammaticality as highly promising for predicting acceptability. However, several aspects of the study presented should be taken cautiously.

[9] Sentences showing a too low reliability were removed.

Shortcomings First of all, the small size of the data sample (16 sentences at most) makes it quite hard to draw definite conclusions regarding the reliability of the g–model, and scaling up the experiments is necessary.

Secondly, very little is said about the input material underpinning the different calculations, such as grammar in use, syntactic parse and characterisation (in the sense of Property Grammars) of the input strings. Unfortunately, the influence of these factors on the final score is not discussed. The content of the characterisation, for instance, plays a crucial role and is directly and tightly linked to various choices made concerning the syntactic structure under consideration. We also notice that the weighting scheme provided p. 61 of BHR06 does not include any *dependency* constraints, even though *dependency* is listed in §2 (p. 58) as a pre-defined constraint type in PG.

And thirdly, the question of the scalability of the model involves addressing the problem of processing text automatically in order to provide the model with the expected—and required—syntactic material. We understand that the results presented in BHR06 were based on manual characterisations of the corpus being investigated. Such a manual characterisation disguises the problems encountered when characterising (*i.e.* parsing) automatically. The policy adopted for choosing the optimal intermediate parses among all the possible candidates, for instance, is not mentioned.

In addition to the model of GI introduced by Blache *et al.*, we also investigate other models which we have designed. Those new models are defined below.

Coherence

The index of *Coherence* is similar to the one of Grammaticality, except that it relies on Anti-Precision rather than Precision.

Definition 5.9 (Coherence) *We define the* Coherence *of a constituent c recursively as follows:*

$$\gamma_c = \widetilde{\mathcal{P}}_c \cdot \overline{\gamma_{c_i}} = \widetilde{\mathcal{P}}_c \cdot \frac{\sum_{i=1}^{Z_c} \gamma_{c_i}}{Z_c}$$

Taxed Coherence

The index of *Taxed Coherence* is again similar to the one of Grammaticality, except that Precision is replaced by Taxed Precision.

Definition 5.10 (Taxed Coherence) *We define the* Taxed Coherence *of a constituent c recursively as follows:*

$$\gamma'_c = \widetilde{\mathcal{P}'}_c \cdot \overline{\gamma'_{c_i}} = \widetilde{\mathcal{P}'}_c \cdot \frac{\sum_{i=1}^{Z_c} \gamma'_{c_i}}{Z_c}$$

5.5 Empirical Investigation

In this section we propose to investigate comparatively the three models for syntactic gradience we have just specified. Each of these models is defined for the same nine variable parameters: six of them are the weights assigned to each of the six constraint types, and three of them are multiplicative coefficients for the different terms of the scoring functions. Since the goal assigned to these models is to predict human acceptability judgements, our task, in this experiment, is to seek the combination of values for the different variables in such a way that the scores computed by the models correlate as well as possible with the reference values of acceptability. The problem is one of *optimisation*, which involves choosing values for the nine different variables available. The *parameter space* is, therefore, a 9-dimension space.

Two questions thus arise regarding how to linguistically interpret different parts of the parameter space:

1. What kind of correlation are we aiming for? That is, what constitutes a good correlation?

2. Is there a linguistic interpretation of the values assigned to the different parameters?

Question 1 is equivalent to asking (i) whether we are aiming to *minimise* or *maximise* the correlation, and (ii) what is a linguistic interpretation of the value for the correlation coefficient.

As far as predicting acceptability is concerned, the greater the (reference) score, the more acceptable the (type of) utterance. Therefore, we are aiming to is maximise the scores computed by the models. Since no restriction is put yet on the values to be assigned to parameters, the possibility of negative correlations must be considered. The interpretation of a negative correlation coefficient would be that the more acceptable a type of utterance, the less acceptable the prediction by the model. Therefore an anti-correlation (*i.e.* a Pearson's correlation coefficient $\rho < 0$) is not linguistically motivated, and the corresponding part of the parameter space must be discarded. The model's score can always be negated in order to obtain $\rho > 0$ and discard the parameter space corresponding to $\rho < 0$.

Question 2 relates to determining a domain of value for each variable, and led us to restrict it. Six of the parameters are weights assigned to constraint types, which are meant to capture the impact of each constraint type on acceptability in absolute value[10]. Therefore, negative values carry no meaning for them and are subsequently discarded. The other three parameters are adjustments of the tradeoff among the different individual scores involved in each model. These adjustments inform on the relative influence of each of the postulates captured by the individual scores on the overall rating. Different elements of linguistic interpretation of all these parameters are discussed along §§5.5.1 to 5.5.4, on the basis of the numeric results we present.

This optimisation task was performed manually, using the method of the *gradient ascent*. We use as a starting point for the set of parameter values those reported by BHR06 as performing best for the g–model. We first choose one dimension (*i.e.* one parameter) and seek an improvement of the correlation coefficient in increasing the parameter progressively, until a local maximum is reached. We iterate the process in each direction of the parameter space individually, until we can conjecture a ranking of all the parameters. We then seek further improvement of the correlation in modifying the order of magnitude for each parameter value. Different arbitrary changes are also made in the ranking of the parameter values on the basis of the change of correlation observed, in order to make sure that another local maximum can not be reached. It

[10] Whether a constraint type has a positive or a negative influence on gradience is not captured by a signed value of its weight but rather through the relative importance of this weight with respect to the other ones.

would be interesting in further works to think about how to automate these arbitrary changes, in a *Simulated Annealing* fashion.

5.5.1 Model Calibration

Calibrating the models consists of finding the right order of magnitude among the different parameters (*i.e.* adjustment coefficients and constraint weights) in order to maximise the correlation with the values of acceptability as established by human judgement. Different combinations were attempted, which are reported in Table 5.3, along with the correlations obtained. A *record* in the table is a run of *Numbat* over the entire corpus with all the parameters being set to the corresponding values (adjustments and weights).

	Adjust.			Weight						Correlation			
$\sharp$	k	l	m	wl	wo	we	wr	wd	wu	Max	g	γ	γ'
1	10	6	2	5	3	1	1	1	2	0.2767	0.0926	0.2747	***0.2767**
2	1	6	3	10	4	3	3	2	1	0.3071	0.2073	0.1308	***0.3071**
3	10	4	1	15	5	1	1	1	1	0.3429	0.2141	***0.3429**	0.3295
4	7	3	2	10	5	1	2	1	3	0.4193	0.2418	0.4156	***0.4193**
5	5	10	8	20	10	4	2	1	2	0.4315	0.0779	0.3998	***0.4315**
6	5	10	2	4	3	1	1	1	2	0.4317	0.0762	***0.4317**	0.4041
7	5	2	1	5	3	2	2	1	2	0.4672	0.3371	0.4634	***0.4672**
8	4	2	1	20	3	5	4	2	10	0.4707	0.3932	0.4658	***0.4707**
9	6	2	1	10	5	2	3	1	2	0.4767	0.3664	***0.4767**	0.4746
10	4	2	1	10	7	2	3	4	5	0.4918	0.3609	0.4845	***0.4918**
11	4	2	1	5	3	2	2	0	2	0.4974	0.3891	0.4945	***0.4974**
12	4	2	1	5	3	2	2	1	2	0.4984	0.3805	0.4946	***0.4984**
13	5	2	1	10	5	2	2	1	2	0.5145	0.4153	***0.5145**	0.5121
14	4	2	1	20	10	4	4	2	4	0.5340	0.4529	***0.5340**	0.5325
15	4	2	1	20	10	2	3	4	5	0.5350	0.4402	0.5342	***0.5350**
16	4	2	1	20	10	2	4	3	1	0.5402	0.4857	***0.5402**	0.5345
17	4	2	1	20	10	5	4	3	2	0.5425	0.4745	***0.5425**	0.5381
										0.5425			

Table 5.3: Calibration of combined adjustments and constraint weights for all three models. *Column* $\sharp$ *is a record Id; column* Max *contains the maximum correlation for each record—the corresponding value is also pointed out visually in the* Correlation *columns* (***0.000**)*; the records are sorted by increasing* Max *correlation.*

Observations and Speculations

1. The best correlation ($\rho = 0.5425$) is obtained for record ♯17, for the γ–model. The constraint weights are ranked in the following order: $wu < wd < wr < we < wo < wl$, that is, *uniqueness* < *dependency* < *requirement* < *exclusion* < *obligation* < *linearity*. It is important to emphasise that this constraint ranking differs from the one in Keller (2000) in that the constraints are not only ranked according to how much unacceptability they entail, as in Keller, but according to how important they are in *absolute value* with respect to acceptability.

2. Whatever the combination of parameters, γ and γ' always perform better than g, which confirms, as suggested in §5.4.2 when defining the Anti-Precision score, that the g–model is over-emphasising the role of success cumulativity compared to the role of failure cumulativity. g is even outperformed on records ♯11 and ♯12, which reproduce the set of values reported in BHR06 as the best performing one.

3. The best performing set of values (rec. ♯17) grants much importance to Linearity (a factor 10 between $wl = 20$ and the minimum $wu = 2$, and a factor 2 between wl and its next follower $wo = 10$), then to Obligation (a factor 5 between $wo = 10$ and wu, and a factor 2 between wo and its very next follower $we = 5$). Then follow the remaining weights, ranging over $[2 \ldots 5]$. This observation of two constraint types (namely Linearity and Obligation) having relatively high weights on the one hand, and the remaining weights having rather low values on the other hand, seems to confirm the hard *vs.* soft dichotomy discussed in §2.4.2.

5.5.2 Data Sample from Blache *et al.* (2006)

In order to perform a more accurate comparison between our results using the parser's outcome and the results from the semi-automatic experiments reported in BHR06 we performed a series of measurements using the same data sample as Blache et *al.*, which is a subset of the full corpus. The results are reported in Table 5.4, and the scatter plot from Figure 5.3 illustrates how the g–model fits acceptability. The correlation coefficient ($\rho = 0.6427$) obtained is much better than the one over the full corpus. Blache and colleagues report an even better correlation coefficient than ours of 0.76. However, their experiment is semi-automatic, and relies on characterisations of the sentences which were performed manually, therefore avoiding parsing problems.

	Adjust.			Weight						Correlation			
♯	k	l	m	wl	wo	we	wr	wd	wu	Max	g	γ	γ'
1	4	2	1	20	3	5	4	2	10	0.5301	0.5294	***0.5301**	0.5193
2	4	2	1	5	3	2	2	0	2	0.6017	0.5408	***0.6017**	0.5903
3	4	2	1	5	3	2	2	1	2	0.6017	0.5246	***0.6017**	0.5925
4	4	2	1	20	10	5	4	3	2	0.6427	***0.6427**	0.6024	0.5872
										0.6427			

Table 5.4: Correlations on the same data sample as in Blache, Hemforth, and Rauzy (2006). *The model discussed by BHR06 is g; Record ♯2 and ♯3 use the same parameter scheme as the best one reported in BHR06 (Record ♯2 simulates the absence of dependency constraints); Record ♯4 uses the same scheme as the best one from Table 5.3.*

The best correlation (rec. ♯4) is obtained for the same scheme as the best one from Table 5.3, but surprisingly this time g outperforms the other two models. This emphasises the crucial influence of Linearity on acceptability, but the roles of Uniqueness and Obligation are still unclear – though they are seemingly preponderant.

5.5.3 Top Scores

Model Fits

Figures 5.4 to 5.6 illustrate how each model fits acceptability judgement for their respective top scoring correlation. All three scatter plots show a normal distribution. Nevertheless, the figures reveal pairs of correlates which lie significantly far apart from the rest of the distribution.

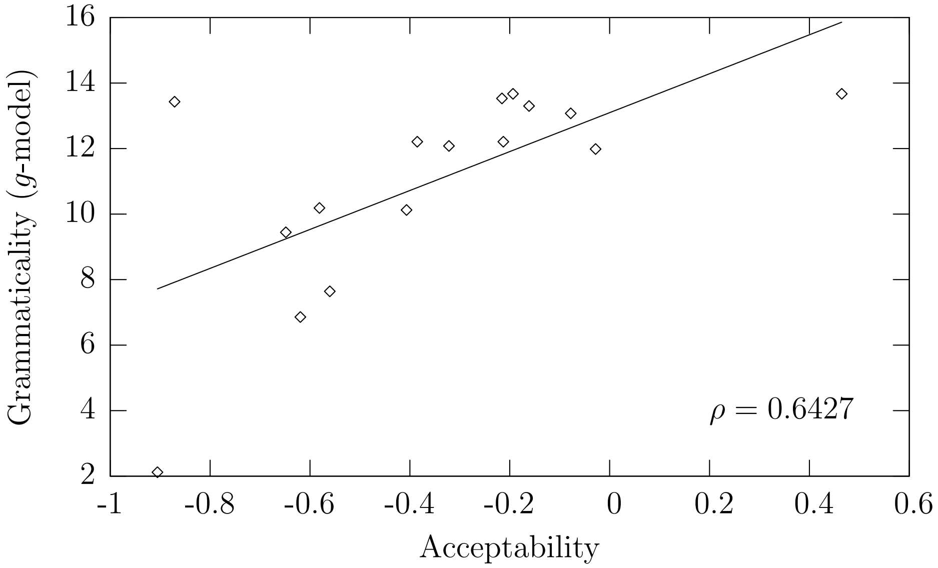


Figure 5.3: Grammaticality (Index) *vs.* Acceptability: model fit over the same data sample as in BHR06.

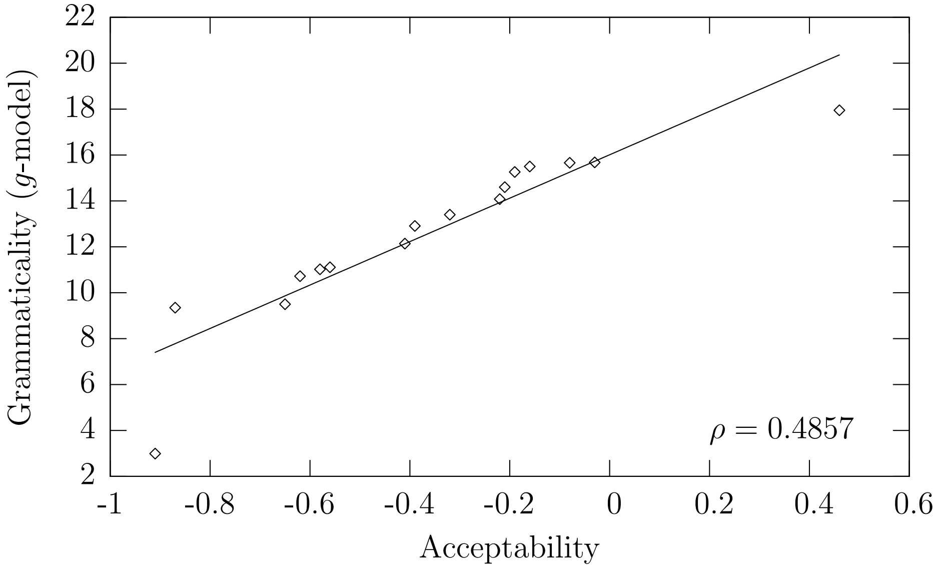


Figure 5.4: Grammaticality (Index) *vs.* Acceptability: model fit over the full corpus

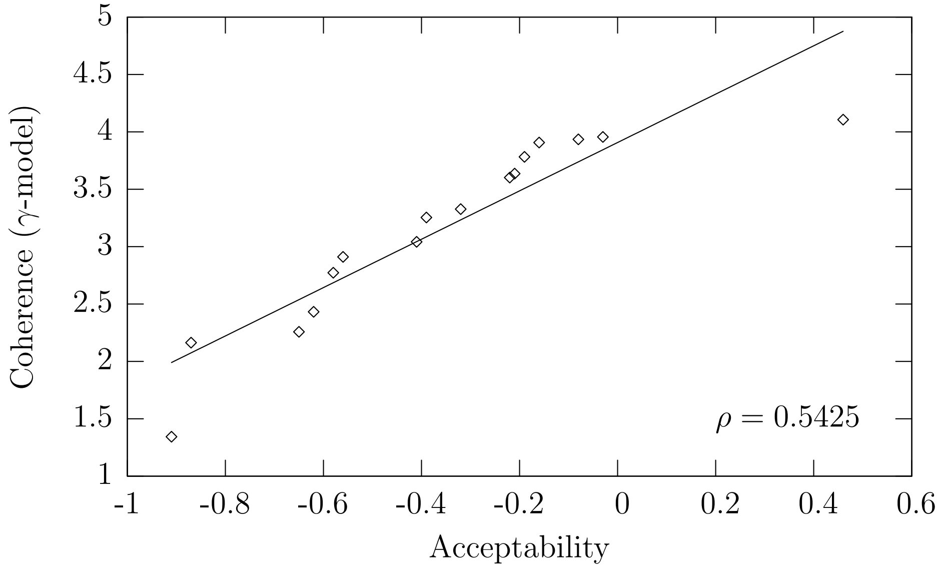


Figure 5.5: Coherence *vs.* Acceptability: model fit over the full corpus

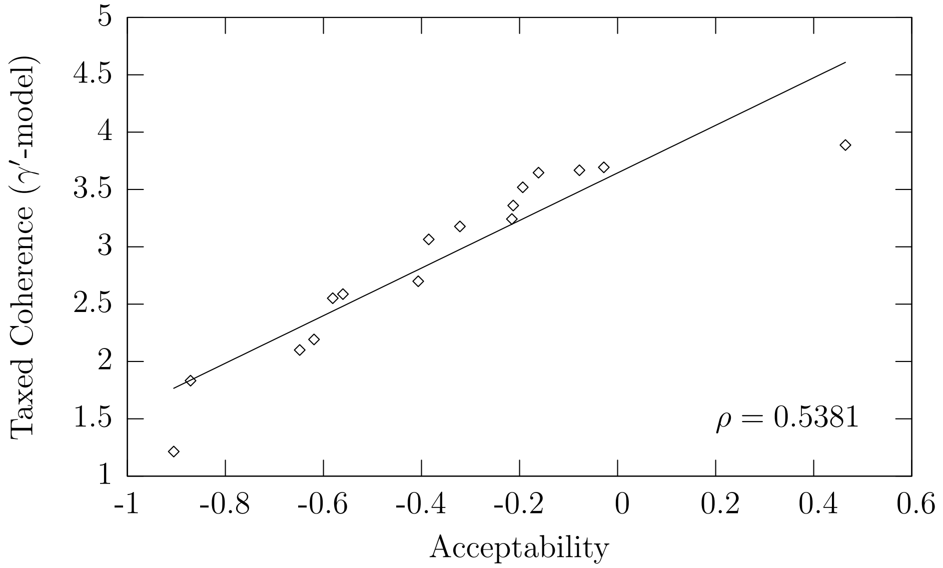


Figure 5.6: Taxed Coherence *vs.* Acceptability: model fit over the full corpus

Gradients

Now, if we look in more detail at how the models rate the different sentence types a gradient can be established for each model. These gradients are reported in Tables 5.6 to 5.8, where the detailed scores are sorted in decreasing order (best is first). These figures are those obtained by every model in its best configuration—*i.e.*, g is parametered as in record ♯16 from Table 5.3, and γ and γ' are parametered as in ♯17. As a reminder, Table 5.5 reports the reference scores of acceptability (human judgement) for each type of sentence, sorted in decreasing order as well. The column "Ref. Rank" in each table indicates the rank of the score given by the human judges for this type.

Rank	Type	Acceptability
1	1.1	0.4647
2	2.4	-0.0281
3	4.2	-0.0779
4	2.2	-0.1614
5	5.2	-0.1935
6	4.3	-0.2129
7	3.2	-0.2157
8	5.4	-0.3217
9	4.4	-0.3852
10	3.1	-0.4067
11	5.1	-0.5603
12	4.1	-0.5811
13	3.3	-0.6193
14	2.1	-0.6484
15	2.3	-0.8711
16	5.3	-0.9054

Table 5.5: Human judgements of acceptability – reference scores, sorted from best (♯1) to worst (♯16). *Each score is the mean normalised score for the sentence type; each type of sentence corresponds to a specific error pattern; the individual scores were obtained by Magnitude Estimation.*

Type	Ref. Rank	Rank	g
2.3	15	1	17.9491
1.1	1	2	15.6753
5.2	5	3	15.6594
3.2	7	4	15.5030
4.2	3	5	15.2604
3.1	10	6	14.5974
4.4	9	7	14.0848
2.2	4	8	13.3994
4.3	6	9	12.9084
2.4	2	10	12.1384
4.1	12	11	11.1118
5.4	8	12	11.0152
3.3	13	13	10.7176
5.1	11	14	9.5038
2.1	14	15	9.3499
5.3	16	16	2.9866

Table 5.6: Gradient of g–scores, sorted from best (♯1) to worst (♯16).

Type	Ref. Rank	Rank	γ
2.3	15	1	4.1063
1.1	1	2	3.9557
5.2	5	3	3.9347
3.2	7	4	3.9068
4.2	3	5	3.7824
3.1	10	6	3.6361
2.2	4	7	3.6019
4.3	6	8	3.3271
2.4	2	9	3.2541
3.3	13	10	3.0417
5.4	8	11	2.9105
4.1	12	12	2.7727
2.1	14	13	2.4321
5.1	11	14	2.2578
4.4	9	15	2.1632
5.3	16	16	1.3439

Table 5.7: Gradient of γ–scores, sorted from best (♯1) to worst (♯16).

Type	Ref. Rank	Rank	γ'
2.3	15	1	3.8873
1.1	1	2	3.6930
5.2	5	3	3.6674
3.2	7	4	3.6467
4.2	3	5	3.5193
2.2	4	6	3.3601
3.1	10	7	3.2425
4.3	6	8	3.1775
2.4	2	9	3.0647
3.3	13	10	2.6997
4.1	12	11	2.5873
5.4	8	12	2.5517
2.1	14	13	2.1923
5.1	11	14	2.1000
4.4	9	15	1.8326
5.3	16	16	1.2145

Table 5.8: Gradient of γ'–scores, sorted from best (♯1) to worst (♯16).

5.5.4 Interpretation

In this section we interpret some significant figures observed above.

As a preamble, we observe that the gradients for the different models reported in Tables 5.6 to 5.8 all present a reasonable match to the reference gradient. The most important discrepancies concern the Types 2.3, 2.4, and 4.4. Type 5.3 also shows interesting characteristics, as will presently be explained.

VP-violation: Type 5.3

Type 5.3 (VP-violation), which comes worst by human standards, is ranked correctly at ♯16 by all models (see Tables 5.6, 5.7 and 5.8). However, the scatter plots from Figures 5.4 to 5.6 also show that the corresponding pair of correlates is always way out of range (bottom left corner). This is explained by the fact that all parses of

Type 5.3 are always forests of partial parses[11] (with only one exception). Cases of parses for Type 5.3 are exemplified in Figures 5.7, 5.8 and 5.9 (see Table 3.1 for a legend). Sentences of Type 5 concern VP violations; more specifically, those of Type 5.3 are missing head verbs. As already discussed in §4.9, this is a case where the parsing strategy shows weaknesses. Although the strategy in itself does not grant any preferred role to head verbs in VP, the grammar, nevertheless, does, since 77% of the specification of VP are constraints which concern a verb. Therefore, finding a VP in a sentence without any verb in only using the constraints that hold and fail is extremely unlikely, since such a sentence is only concerned by at most 23% of the VP specification. The same problem is propagated in cascade at the sentence level.

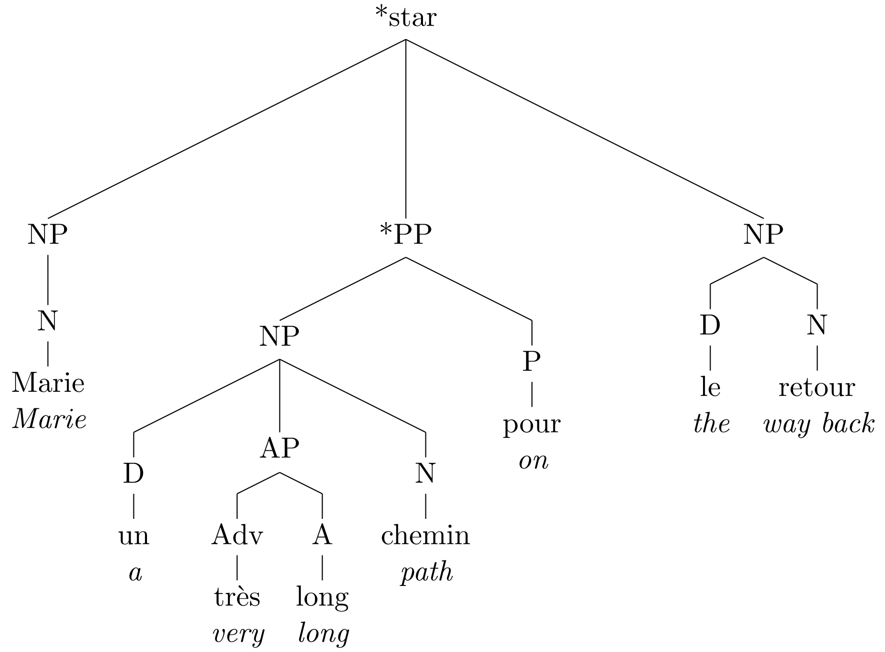


Figure 5.7: Example of a forest of partial parses automatically generated by *Numbat* for Type 5.3 (VP-violation)—*Mary a very long path on the way back.*

These cases reveal the relative weakness of the model to accurately account for forests of partial parses: it successfully ranks them, but the score they are assigned is much too low compared to the others.

[11] Type 5.3 is the only error pattern for which it happens.

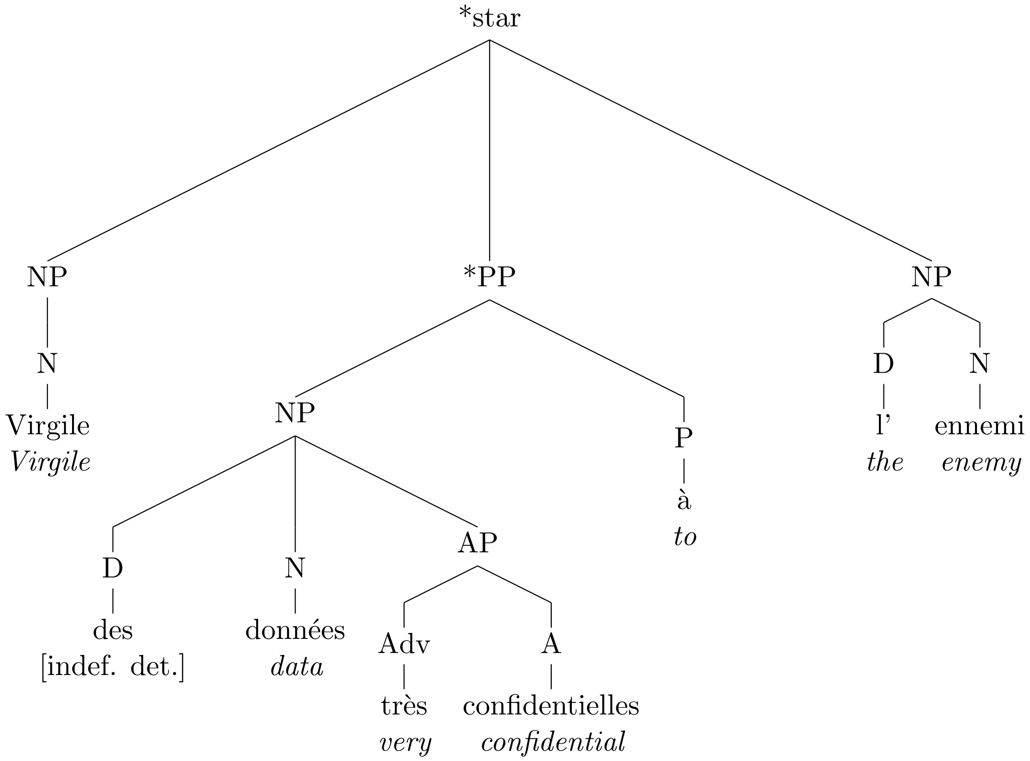


Figure 5.8: Example of a forest of partial parses automatically generated by *Numbat* for Type 5.3 (VP-violation)—*Virgile very confidential data to the enemy.*

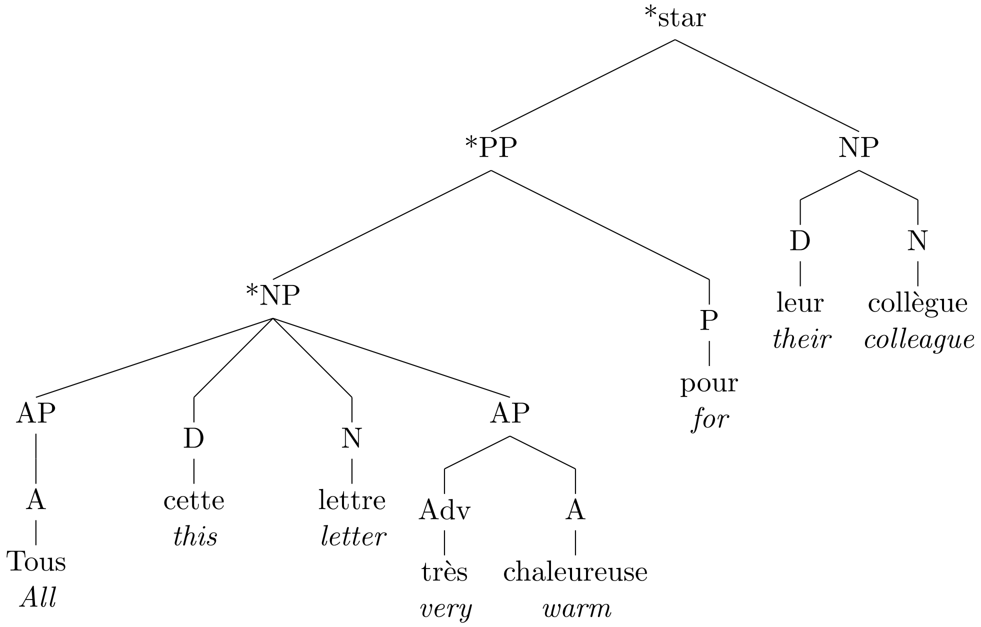


Figure 5.9: Example of a forest of partial parses automatically generated by *Numbat* for Type 5.3 (VP-violation)—*All this very warm letter for their colleague.*

NP-violation: Type 2.3

The sentence Type ranked highest is the same for all models (2.3, NP-violation), whereas it is ranked second last by the human judges, *i.e.* ♯15. This type concerns a missing head noun, exemplified in Figures 5.10 and 5.11 (see Table 3.1 for a legend).

Such a discrepancy is explained by a combination of elements. One is the same as the one discussed about Type 5.3, concerning the preponderant role of heads in constructions such as NP and VP, which are specified by a very large number of constraints compared to other constructions. The same negative effect is observed here, where the absence of head noun prevents *Numbat* from finding a noun phrase. However, a second explanation exists for the case of the NP specification, which unlike the one for VP allows either an N *or* an AP (*i.e.* a substantive adjective) to head the construction. This explains why sentences such as those in Figures 5.10 and 5.11 are parsed correctly. This alternative for the head position of NP exists in French

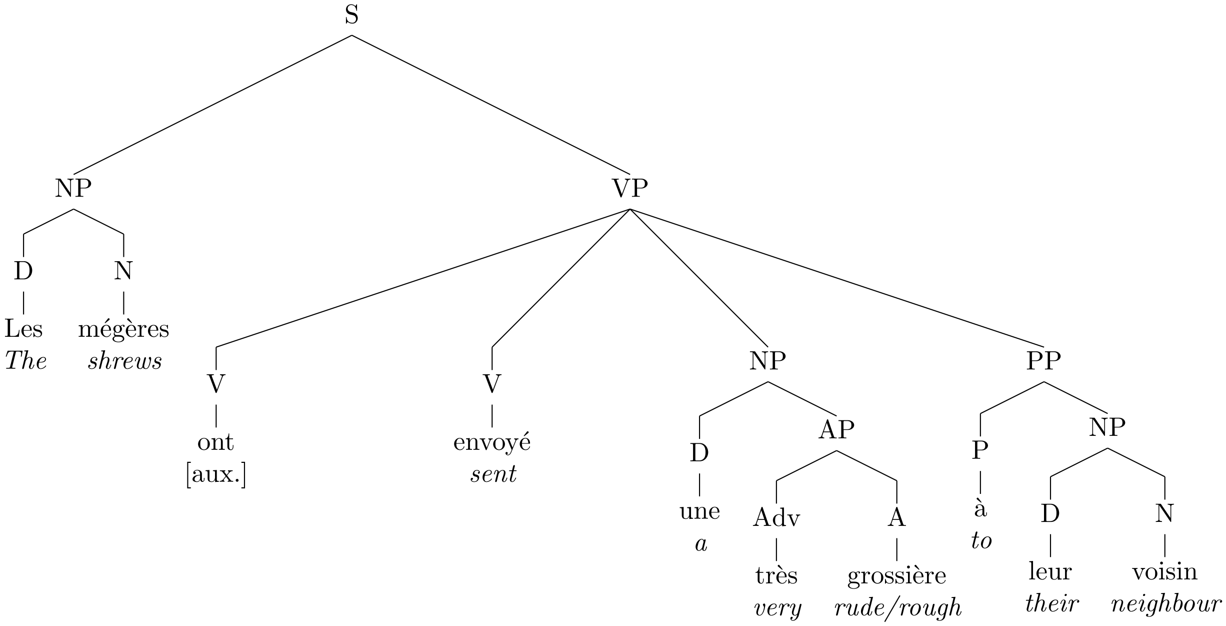


Figure 5.10: Example of a parse automatically generated by *Numbat* for Type 2.3 (NP-violation)—*The shrews* [aux.] *sent a very rude/rough to their neighbour.*

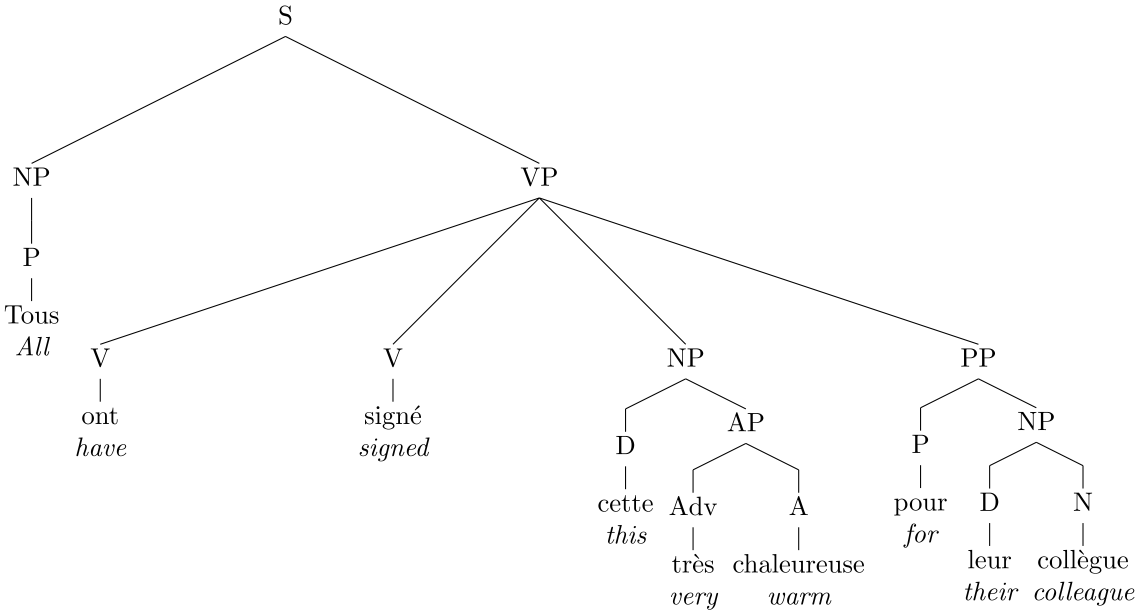


Figure 5.11: Example of a parse automatically generated by *Numbat* for Type 2.3 (NP-violation)—*All have signed this very warm for their colleague.*

in order to license perfectly fine cases of substantive adjectives, such as in (5.1) and (5.2)—the corresponding parses are illustrated in Figure 5.12.

(5.1) Je lis le petit.
I read the small [one].

(5.2) Marie emporte le froid pour midi.
Marie takes the cold [one] for lunch.

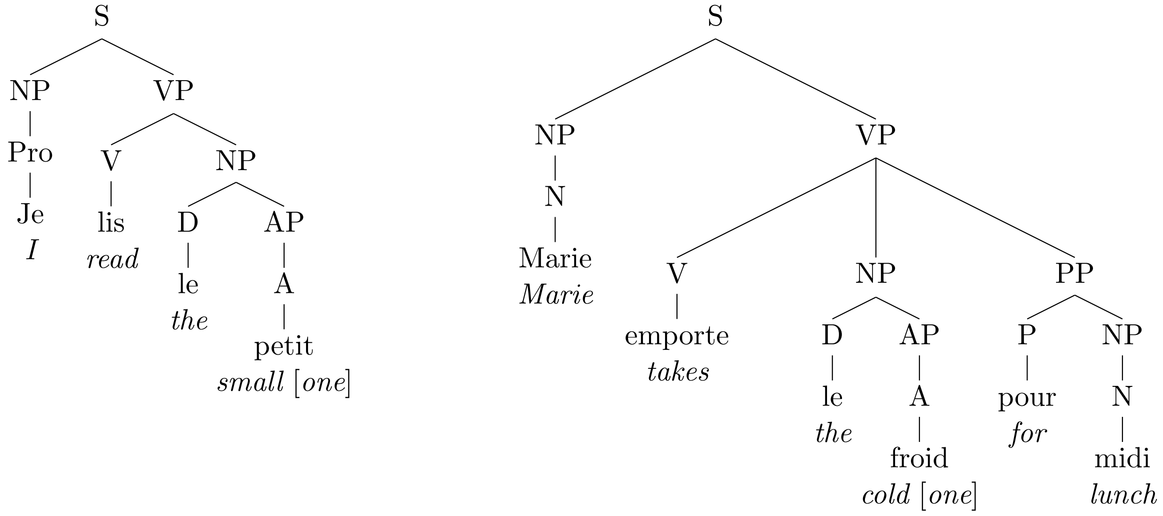


Figure 5.12: Substantive adjectives: constituent structures for the example sentences (5.1) and (5.2). *Parses generated manually.*

These cases are discussed in §4.9.4. They were meant to be deviant as a result of having an error pattern applied to them, but their deviance turns out to be arguable.

Two conclusions may be drawn from the problem raised by these cases of NP-violation (Type 2.3). One conclusion concerns ambiguities in the grammar, which the parser can hardly be expected to clear up. These ambiguities mostly come from that the grammar is clearly under-specified. One option could have consisted in translating the lexicalised Tree-Adjoining Grammar (TAG) for French (FTAG) developed by Abeillé and Candito (2000) into a Property Grammar. The problem, then, would be to work out translation rules from TAG to PG. Whether achieved automatically or manually the task appears to us as a non-trivial one. And since FTAG was, to the best of our knowledge, the only[12] large coverage grammar readily available for French at the time the experiments were run, we chose to give our work a different direction. Incidently, it is worth noticing that the source of ambiguity does not exclusively come from the grammar itself. The ability of a lexicon to encode more fine-grained information, or the ability of a POS-tagger to handle it, could also be blamed to some extent.

Another conclusion concerns the limited content of the corpus—or, should we say, the lack of a larger one annotated with human judgement of acceptability. It would be interesting, for instance, to investigate cases of NP-violation similar to Type 2.3 where sentences contain no AP in a position to be taken as a substantive one.

NP-violation: Type 2.4

Sentences of Type 2.4 (missing determiner in NP) are ranked second most acceptable by humans, while all the models rank them ♯9 or ♯10. Figure 5.5.4 shows that the optimal parse generated by *Numbat* for this error pattern successfully finds an NP where the determiner is missing. This case looks a bit puzzling to us. What we can say is that we observe that for the three Types 5.3, 2.3, and 2.4 which raise discrepancies, a missing construction is concerned: missing head verb for 5.3, missing head noun for 2.3, and now missing determiner for 2.4. It is likely that what these cases put forward is a weakness of all three numeric models to accurately report on a missing element, or maybe on a missing requirement.

[12] In her PhD thesis Guénot (2006) presents a grammar “covering most of the basic syntactic phenomena for French”, which was developed in the framework of PG. By the time the grammar was published, integrating it with ours turned out to involve complex and time-consuming adaptations of both the *Numbat* parser and Guénot’s grammar, in order to overcome various technical issues and incompatibilities. Among other reasons, the feature structure used by Guénot, for instance, is much more detailed and fine-grained than the one we have implemented in *Numbat*; the extensive use of disjunctions in different constraints is also much broader than what *Numbat* can handle and require de-multiplying these constraints into more basic ones; Guénot’s grammar also relies on inheritance mechanisms among constructions, which are not all implemented in *Numbat*.

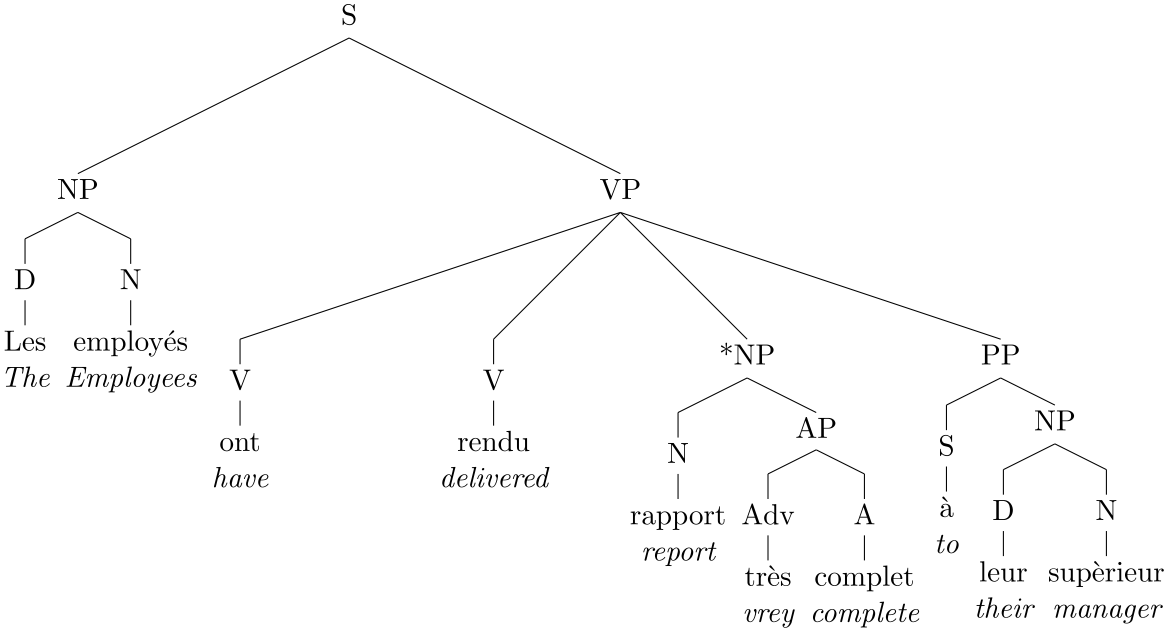


Figure 5.13: Example of a parse automatically generated by *Numbat* for Type 2.4 (NP-violation: missing determiner)—*The employees have delivered very complete report to their manager.*

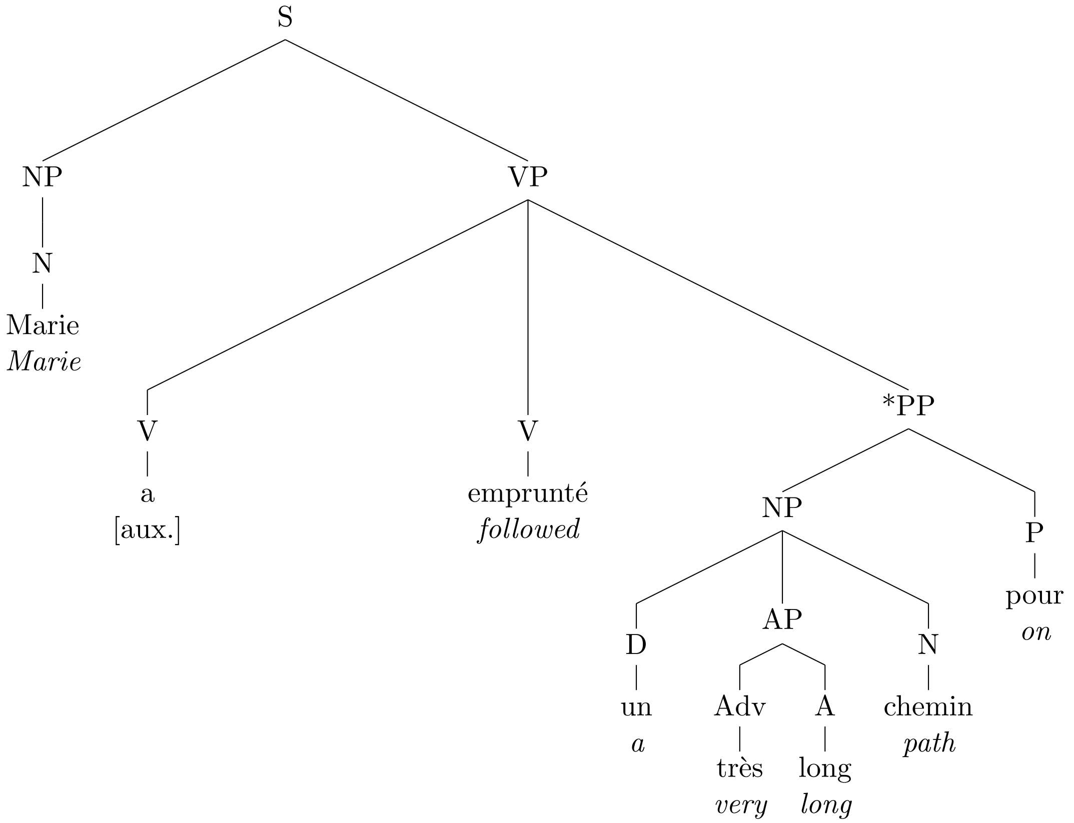


Figure 5.14: Example of a parse automatically generated by *Numbat* for Type 4.4 (PP-violation)—*Marie* [aux.] *followed a very long path on.*

PP-violation: Type 4.4

Sentences of Type 4.4 (PP-violation) are another case of divergent correlates. Sentences of this type—exemplified in Figures 5.14 and 5.16—end ab-normally with a preposition. *Numbat* treats many cases of PP-violation of Type 4.4 as of word order violations and groups the final preposition with the preceding NP in order to make a PP. This does not capture well enough a reader's intuition that the final preposition is expected to introduce an upcoming PP—that is, the expectation is that a noun phrase should follow the preposition. Figure 5.15 shows that even in the sub-optimal alternative solutions *Numbat* tries to group the final Preposition with what precedes

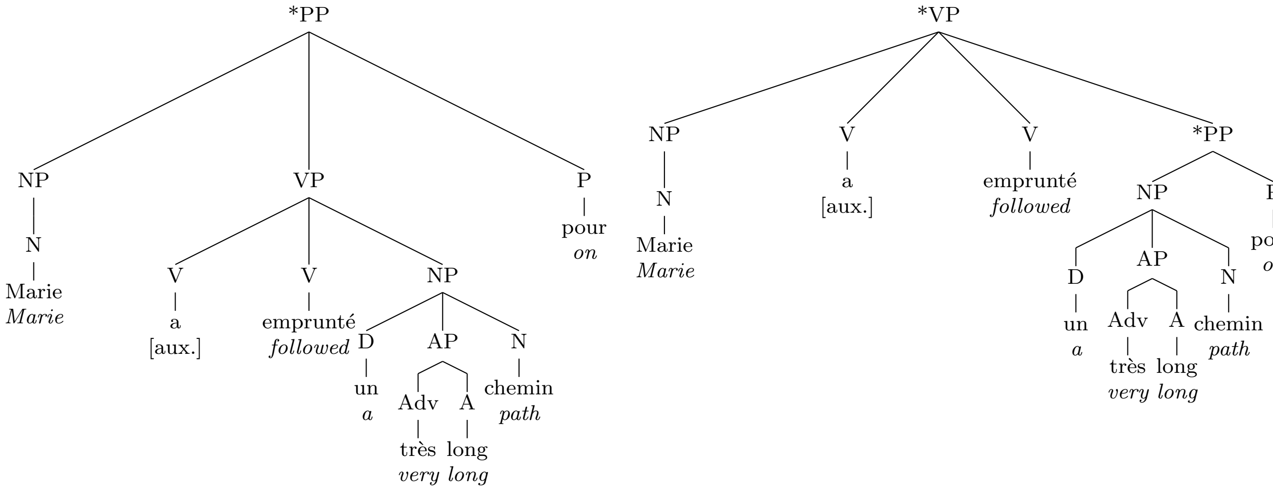


Figure 5.15: Alternative parses automatically generated by *Numbat* for the same input as in Figure 5.14.

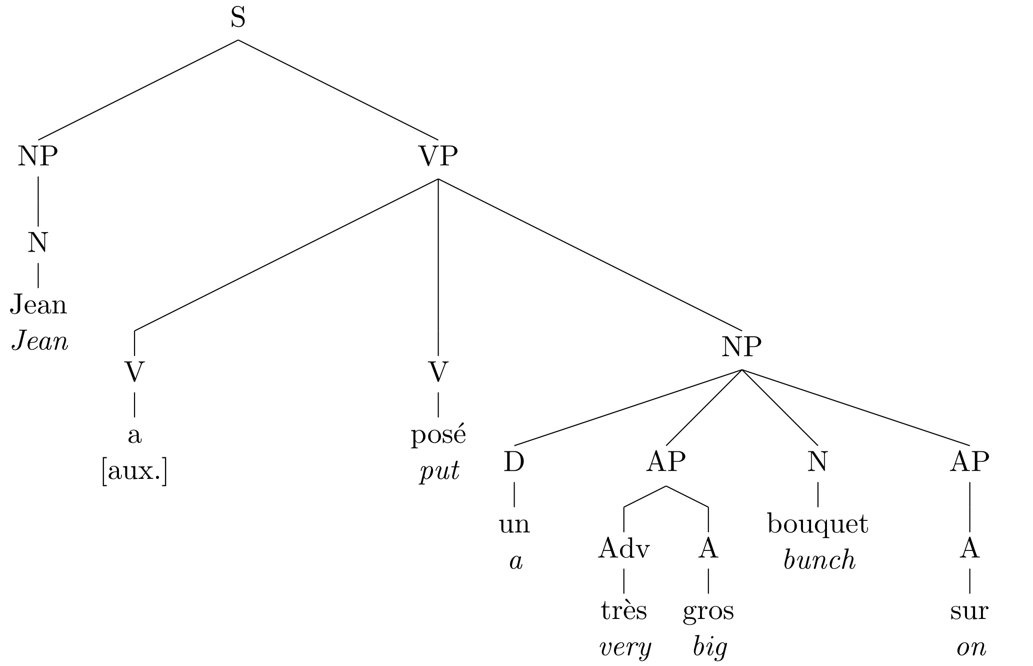


Figure 5.16: Example of a parse automatically generated by *Numbat* for Type 4.4 (PP-violation)—*Jean* [aux.] *put a very big bunch on.* Note that the English translation does not accurately reflect the error pattern, since *to put on* could be seen as a phrasal verb; using *on top of* instead of *on* would better show the missing end of the sentence, but is no longer a preposition.

it. This can be explained by the fact that *Numbat* aims to minimise the proportion of violated constraints. Therefore, a violation of word order with preceding constituents is very likely to always be preferred over than alternative options which would have to consider hypothetical missing constituents. This type of error pattern raises a limitation of the strategy of optimality adopted in *Numbat*.

5.6 Conclusion

In this chapter we have introduced two models (γ and γ') for a computational account of syntactic gradience. These models rate a natural language utterance according to its syntactic characterisation.

We have postulated that acceptability can be predicted by factors derivable from the output of the LSCP. Three of these factors ((i) to (iii) below) are substantiated by empirical evidence and work from linguistics and psycholinguistics, while two others ((iv) and (v) below) are hypothetical. These factors are the following:

(i) *failure cumulativity* stipulates that the effects of failing constraints cumulate towards lower acceptability;

(ii) *success cumulativity* stipulates that successful constraints tend to induce a convergence towards higher acceptability;

(iii) *constraint ranking* stipulates that all constraints be weighted proportionally to their impact on acceptability, but does not presume exact values for the weighting scheme which is fine-tuned empirically;

(iv) *recursive cumulativity* stipulates that nested constituents recursively influence acceptability of their mother constituent;

(v) *constructional complexity* stipulates that acceptability is influenced by the relative complexity of constituent types, measured by the number of constraints required to specify them. The models we have proposed implement these five factors in various ways.

We have run a comparative empirical investigation of the two models we specified, along with a third pre-existing one presented in BHR06, in order to determine how their respective ratings correlate with human judgement of acceptability. Each of the three models is only relying on the syntactic characterisation provided by *Numbat*. The experiments were performed over a 512–sentence (about 6000 words) corpus in which the sentences are evenly spread among 15 types of deviance and 1 type of well-formedness. This corpus was annotated with human judgements of acceptability, as a result of psycholinguistic works carried out independently from the present study and are reported in BHR06. The best correlation ($\rho = 0.5425$) was found for one of the two models we introduced, namely the index of Coherence (γ–model). The fact that the model outperforms the one of BHR06 (g–model) confirms the hypothesis that the Grammaticality Index over-emphasises the role of Success Cumulativity compared to the one of Failure Cumulativity. This result could also be seen, to some extent, as a corroboration of Keller and the LOT's emphasis on violation; the tradeoff, though, in the g–model, between satisfaction and violation remains significant, whereas it is absent from LOT.

The significance of the results presented in this chapter is tightly linked to the significance of the syntactic material used by the different models. The fact that our experiments were run using the parses generated automatically by *Numbat* is one of the major contributions of this dissertation. When evaluating *Numbat* (see §4.9) we measured a Precision of 0.74 and a recall of 0.68. Although these figures reveal a reasonably good quality of the syntactic material produced by the parser, it also reveals shortcomings regarding how to handle deviant language: 32% of the input strings are still not good enough—or, more precisely, not good enough for the purpose of modelling their syntactic gradience. That is, the correlation of 0.5425 is obtained in spite of 32% of the corpus being characterised incorrectly or insufficiently correctly. Different types of sentences—*i.e.* error patterns—were discussed for being involved in pairs of bad correlates. In most cases the problem was identified as stemming from issues in the syntactic material provided by the parser, with maybe one exception for the sentence Type 2.4. Therefore, we are confident that further improvements of the parser's accuracy should subsequently improve the performance of the different models.

Our investigation of the different models provided a weighting scheme for the constraint types, which was observed empirically. This weighting scheme lets us rank the constraint types, in absolute value, in order of increasing importance for acceptability. The γ–model ranks the constraint types the following order: Uniqueness < Dependency < Requirement < Exclusion < Obligation < Linearity. The higher the constraint is ranked, the more influential it is on acceptability, whether positively when the constraint holds, or negatively when it fails. The order of magnitude among γ–scores along the gradient seems to confirm the *soft* vs. *hard* dichotomy suggested by Keller (2000, 2003), and discriminate *linearity* and *obligation* as hard constraints. However, further work is required in order to confirm and narrow down this classification.

Another contribution from the γ–model is its potential prediction power. We showed that the significantly bad correlates could be explained by issues in the input syntactic characterisation rather than by the γ–rating itself. Cases of false-positives, pseudo-false-negatives, forests of partial parses, and under-specification of the grammar were discussed.

5.6.1 Further Work

More Variable Parameters The small number of variable parameters in each model limits their flexibility by restricting the possibilities of calibration. To overcome this inconvenience, it might be interesting to consider the different *tax rates* from the Taxed Coherence (γ'–model) as as many variable parameters.

Backing the Factors of Influence An important avenue of further investigation would be to substantiate all the factors we are using for predicting gradience with linguistic evidence. A body of work on psycholinguistic on linguistic complexity and sentence processing (Gibson, Schütze, and Salomon, 1996; Gibson, 1998, 2000; Warren and Gibson, 2002; Gruber and Gibson, 2004; Grodner and Gibson, 2005), for instance, seems to constitute a good starting point. The Prediction Locality Theory (Gibson, 1998) and related work provides a framework for measuring sentence complexity on psycholinguistic grounds, which might be able to also provide clues as to how deviant language can best be processed automatically.

Chapter 6

Conclusion

In this work we have addressed part of the problem concerned with replacing the traditional binary notion of grammaticality with intermediate degrees of acceptability in natural language processing. We have focused more specifically on devising, implementing and evaluating a computational model for syntactic gradience. We have presented a fully automated solution, which characterises any well formed or ill formed input sentence, generates an optimal parse for it, then rates the utterance according to its grammatical acceptability. Our new model extends the concepts of *Intersective Gradience* (IG) and *Subsective Gradience* (SG) introduced by Aarts for modelling graded judgements in order to cover deviant language. Under such a model, the problem raised by gradience is to classify an utterance as a member of a specific category according to its syntactic characteristics. IG, in its extended form, is concerned with choosing the most suitable syntactic structure for an utterance among a set of candidates. This problem was addressed in using a criterion of optimality for discriminating the candidates. SG, in its extended form, is concerned with calculating to what extent the chosen syntactic structure is typical from the category at stake. This problem was addressed in defining a numeric model for rating an utterance according to its grammatical acceptability, using the utterance's syntactic characteristics. Our investigation of different frameworks for representing the syntax of natural language shows that these syntactic characteristics, which serve as features for classifying an utterance, can easily be represented in Model-Theoretic Syntax. We chose to work with Property Grammars (PG), a constraint-based formalism defined by Blache (2001), which offers to model the *characterisation* of an utterance in terms of satisfied and violated properties.

Through the development of such a new model of gradience, the main contribution of this work is three-fold.
First, we specify a model-theoretic logical framework for PG, which bridges the gap observed in the existing formalisation regarding the constraint satisfaction and constraint relaxation mechanisms, and how they relate to the projection of a category during the parsing process. This new framework introduces the notion of *loose satisfaction*, along with a formulation in first-order logic, which enables reasoning about the characterisation of an utterance.
Second, we present our implementation of Loose Satisfaction Chart Parsing (LSCP), a dynamic programming approach based on the above mechanisms, which is proven to always find the full parse of optimal merit.
And third, after postulating that human acceptability judgements can be predicted by factors derivable from LSCP, we present the numeric function of *Coherence* for rating an utterance according to its syntactic gradience. The ability of the model to predict human judgements of grammatical acceptability was investigated. The correlation between Coherence and human acceptability is measured with a coefficient $\rho = 0.54$.

6.1 Summary

We have addressed the problem raised by the development of an automated model of syntactic gradience from the two perspectives of knowledge representation (KR) and numeric modelling. We started by investigating works from the fields of Linguistics and Psycholinguistics analysing gradience in natural language. With respect to KR, we were interested in identifying what sort of linguistic information is involved when dealing with gradience, and how to present, represent, and process it. The literature review revealed that although degrees of acceptability in language are observed and well-acknowledged, traditional approaches to language processing based on Generative-Enumerative Syntax (GES) fail to offer a suitable framework to account for graded aspects of language. Reasoning with gradience requires an ability to represent a description of (morphosyntactic, among others) characteristic properties of uttered language. It requires, in particular, to be able to state characteristic properties for deviant language, which is not possible in GES. On the other hand, the literature review also reveals that Model-Theoretic Syntax (MTS) enables the formulation of such statements about the observed properties of an item.

In Chapter 3, we presented a new logical framework for PG. Our investigation of existing frameworks meeting the requirements we have in terms of KR for modelling gradience led us to conclude that PG is one of the most suitable one. Yet, the formalisation of the paradigm falls short of a formulation for reasoning about the characterisation of an utterance. The projection mechanism, in particular, whereby the construction of a new constituent is inferred from a characterisation during the parsing process, raises different problems. We argued that these problems occur when the projection of a construction is too tightly connected to incremental parsing strategies. It introduces unnecessary limitations in terms of processing, although absent from PG in its original conception. An important argument for introducing a new formulation is that a constructional approach to language processing requires being able to implement a parsing process, which is not driven by any particular type of constraint, such as heads. Constraints in PG all present a property of monotonicity, which raises processing issues with respect to implementing such a no-drive parsing strategy. We showed that the problems subsequent to handling the property of monotonicity occur in the case of an incremental parsing strategy, but can be avoided by further formalising PG. We thus defined a model-theoretic logical framework, which introduces the notion of *loose satisfaction*, in order to capture, independently from any processing aspect, the possibility offered in PG to satisfy and/or relax constraints within the same mechanism. Through loose satisfaction a newly inferred constituent may present anomalies, which make it violate part of the grammar, and still be *loosely consistent*. Through predicate calculus, a constituent structure made up of loosely consistent constituents can then be inferred for any input string. The inference mechanism involved in building a constituent structure allows for both incremental and non-incremental parsing strategies to be implemented.

In Chapter 4, we presented our implementation of Loose Satisfaction Chart Parsing (LSCP) based on the above mechanisms. The LSCP algorithm is a dynamic programming approach, which optimises a merit function. The strategy adopted is non-incremental. The constraint solving process, which yields the constituent structure, does not involve any revision process in order to handle non-monotonic constraints. As such, the strategy is naturally not driven by any of the constraint types in particular—especially not heads. Such a feature is an important one in order, for instance, to allow constructions with no head, as well as for implementing inheritance mechanisms which do not necessarily rely on a *head feature principle* (Guénot, 2006).

Although the algorithm proposed for LSCP shows a high theoretical worst time complexity of $\mathcal{O}(n^4 2^n C^n)$, it is correct by construction and proven to theoretically always find the parse of optimal merit. In practice, the use of heuristics make it perform sufficiently well to allow us experimenting with our models of gradience. In order to evaluate the ability of LSCP to provide full parses for unrestricted input language, the corpus we used was mostly made of controlled deviant utterances (94% of the sentences were ill-formed). 92% of the corpus was parsed with full syntactic structures. The parser's output was evaluated by 6 different human annotators, who were asked to make a binary judgement about the correctness of each parse. The main difficulty of the task was to decide whether the parse structures provided by the parser were the best possible syntactic analysis. Precision (Number of correct trees/Number of complete trees) was measured at 0.74, while Recall (Number of correct trees/Total Number of sentences in the corpus) was measured at 0.68.

In chapter 5, we started by postulating that the acceptability of an utterance can be predicted from the combined effect of different factors of influence. We argued that factors found in existing models presented in the literature, although necessary, were not sufficient to accurately account for all aspects of gradience when taken separately. Subsequently, we have put forward different factors found throughout the literature in order to tackle all of them. As a result, we ended up with five factors of influence:

(i) *failure cumulativity*, often referred to in the literature as *cumulativity effect* (Keller, 2000, 2006; Bresnan and Nikitina, 2003) stipulates that the effects of failing constraints cumulate towards more unacceptability;

(ii) *constraint weighting*, found in many works involving gradience (Blache and Prost, 2005; Blache, Hemforth, and Rauzy, 2006; Bresnan and Nikitina, 2003; Foth, 2004; Gibson, 2000; Heinecke et al., 1998; Keller, 2000; Sorace and Keller, 2005; VanRullen, 2005), stipulates that all constraints may be weighted proportionally to their impact on acceptability;

(iii) *success cumulativity* stipulates, as suggested in Aarts (2004b, 2007) and Blache and Prost (2005); Blache, Hemforth, and Rauzy (2006), that the successful constraints tend to induce a convergence towards more acceptability;

(iv) *propagation* stipulates, as suggested in Blache and Prost (2005), that nested constituents recursively influence acceptability of their mother constituent;

(v) *constructional complexity* stipulates, as suggested in Blache, Hemforth, and Rauzy (2006), that acceptability is influenced by the relative complexity of constituent types, particularly in terms of the amount of constraints required to specify them.

These five factors were numerically captured into scores derived from the parser's output, and combined into two rating models.

The ability of these two models to fit acceptability judgement by human standards was investigated, in comparison with a third pre-existing model. The model of *Coherence* (γ–model), which we devised, turned out to outperform the other two. It shows a good correlation coefficient of 0.5425 with acceptability by human standards. It is especially good when put in the perspective of the parser's evaluation, whose precision and recall of 0.74 an 0.68 respectively show that there is still room for improvement with regard to parsing deviant language with full constituent structures. The interpretation of our experimental results also revealed that the cases for which the model was lacking accuracy at predicting acceptability were cases for which the parser did not provide a correct parse. Such a match between bad performance from the parser and bad performance from the model is also a very encouraging observation, which complements the performance on the positive syntactic structures. Altogether it confirms first, that a numerical model such as the one of Grammaticality or the one of Coherence is a good indicator of the syntactic gradience of an utterance, and second, that rating the syntactic gradience of an utterance provides a good prediction of the degree of acceptability. Furthermore, the match observed for the pairs of bad correlates opens avenues for further works, and lets us speculate on the origin of the problems we identified. Different sources of problems were pointed out, such as the under-specification of the grammar, and of the lexicon.

With respect to the pending questions raised in the Background Chapter (§2.6.2), this dissertation has highlighted different elements of answer.

The notion of Natural Language that we have adopted in our formalisation of PG and in our implementation of LSCP conforms to the intentional one we have expressed initially. It covers expressions, as well as quasi-expressions (in the sense of Pullum), and allows a continuum of acceptability.

The notion of grammaticality that we have adopted and implemented with LSCP combines the binary notion from GES, the degrees of grammaticality and ungrammaticality and the sharp boundaries from the model-theoretic notion, and the degrees of grammaticality and the optimality of a syntactic structure from the optimality-theoretic notion.

The notion of gradience that we have modelled involves a constraint-based characterisation of language, which allows rating an utterance on syntactic criteria according to its degree of acceptability. We have shown that such a modelling of syntactic gradience can be used for predicting human acceptability judgement.

The linguistic knowledge about an utterance that we have adopted is represented in the notion of *characterisation* proposed in PG. It combines traditional constituent structure with a constraint-based representation of the relationships (met or not) among constituents, and allows dealing indifferently with expressions and quasi-expressions.

The notion of constraint that we have implemented is the one offered by PG. We have formalised it using first-order logic with model-theoretic semantics.

The satisfied constraints in our model were granted a competitive role alongside the violated one in accounting for syntactic gradience.

The constraint weights in PG and in LSCP in particular can have various scopes and granularities according to needs. In *Numbat* they have been implemented with the granularity of a constraint type and the scope of the grammar.

Each of the different scoring functions implemented in *Numbat* serves a different purpose and models a specific aspect of syntactic gradience. For the main ones, the Cohesion function rates a candidate structure for an utterance and allows the ranking of candidates according to a criterion of optimality, while the function of Coherence rates an utterance according to its degree of grammaticality or its degree of ungrammaticality.

6.2 Further Work

6.2.1 On Scaling Up

The complexity of the algorithm we have proposed for Loose Satisfaction Chart Parsing, as expected, is proven exponential. This is not surprising from a Constraint Satisfaction Problem—especially not given the solving strategy adopted. One could argue that since such a result was expected, one could have gone for better, well-known alternatives, but the main motivation was to investigate how the way Property Grammars, and Model-Theoretic Syntax more generally, represent information can be used for the purpose of modelling and predicting syntactic gradience, which includes proving the optimality of the solution parse. In that respect, and as far as scaling up the approach is concerned, we observe different elements that deserve some attention.

Regarding the complexity, more than simply a theoretical exercise, calculating it brings some insight for identifying where the risk of explosion comes from. This lets us speculate on which aspects of a knowledge-based parsing strategy for PG should be paid attention to, especially when aiming for generating full phrase structures for unrestricted natural language. Of course, as we said earlier, the strategy that we have implemented is to be blamed, but only to some extent; interestingly enough, with a complexity of $\mathcal{O}(C^n)$ (C being the number of constructions, and n the sentence length in words) the grammar size turns out to be a factor of major influence, through the number of constructions being specified. Intuitively, we know that the grammar size plays a role in a parser's complexity, but its influence is usually not of that order of magnitude. In fact, most of the time it is a negligeable factor against the input length. In LSCP, the factor C^n comes from going through all the configurations of a set partition. Dropping that factor involves reducing either the number of configurations, or reducing the number of set partitions being considered, or preferably both. Reducing the number of configurations will drop the factor C, while reducing the number of set partitions will drop the exponent n. An incremental strategy, for instance, heuristically reduces significantly the number of set partitions, but the number of configurations is likely to remain of the same order of magnitude, since a heuristic in that respect would require speculating on which constructions are worth being considered, and which are worth ignoring. A solution to explore, though, would be to find a way to introduce a top-down prediction in the process. Relying on the probabilities attached to phrase structures in corpora could be an option, if the main focus is put on efficiency rather than on optimality with respect to gradience (the most probable parse is not necessarily the same as the optimal one).

Regarding the incremental strategies, we note that while VanRullen does not calculate the theoretical complexity of his SeedParser, he measures a practical one in correlating the number of instructions involved in parsing a sentence with the sentence length in number of words. The measured complexity is polynomial of degree 2.3 ($C \cdot N^{2.3}$ with C a constant and N the number of words). Yet varying the grammar size is not being considered, so we do not know how it performs in that respect.

In conclusion, and as far as scaling up LSCP is concerned, the question of the grammar size thus becomes a crucial one. Our implementation of an algorithm for LSCP is likely not to be, as such, suitable for large-scale applications involving a large-coverage grammar. Yet, it does not mean that the underpinning strategy is completely void for that matter, but more that it is faced with similar problems to those faced by other Constraint Solving strategies. The difference stands in that the combinatorial explosion is just postponed to later by strategies which do not aim to generate optimal full phrase structures for both well-formed and ill-formed sentences, as we do. And last, the decision to go for a strategy based on an exhaustive search was motivated by the need to show the optimality of the resulting output in order to experiment with gradience and rule out, as much as possible, the possibility of better parses. Scaling up would require revising this aim of optimality.

6.2.2 On Modelling Gradience

How would the results presented in this work scale up for a large-coverage grammar, over non-artificial sentences? The question of the large grammar was already addressed. As for non-artificial sentences, one may also wonder how the model of gradience would behave. When analysing our experimental results with modelling gradience we could not find any substantial evidences that would challenge the very approach of using a numerical model based on constraint counts for predicting gradience. Most of the problems that we have identified drew attention on the parser's output and led us to discuss expectations and content-related requirements in that regard. But this pertains to the model input, not to either the existence of the model itself, or even its scalability. Therefore, it can be argued that the model can be refined in different ways, some of which we will discuss later. We could not think of any strong reasons why it would not scale up successfully, especially over non-artificial sentences. This being said, scaling up the experiments would raise different issues, such as regarding the gathering of human judgements of acceptability, or the model calibration. But assuming these, then a reference scale could be established for gradience scores.

Now, on a different but just as relevant matter, it is clear that acceptability judgement as a whole is more than likely to involve other criteria than just syntactic ones[1]. Therefore, one avenue of investigation for improving and scaling up our model of gradience involves the introduction of criteria taken on other linguistic dimensions than syntax, that is, from semantics, pragmatics, phonology, and so on. Construction

[1] This, of course, does not make the study of syntactic gradience taken separately from other linguistic dimensions irrelevant. Syntactic gradience, even in isolation from the rest, is still informative of the overall acceptability of a sentence.

Grammar conceptually already allows an account of semantic constraints in the specification of constructions, and Property Grammars (PG) have no restriction in this regard either. In chapter 2 we have argued that ruling out semantic constraints from a model of gradience is an impediment to handling constructions such as the English Caused-Motion Construction (EC-MCx), which involve semantic constraints in their specification—*nonstative verb* and *directional phrase* for that matter. One could easily imagine including, in the specification of the EC-MCx within PG, constraints of Obligation over a Verb feature (for *nonstative*), and over a feature of the prepositional phrase inherited from the preposition stating the *directional* semantic aspect. Such a descriptive approach to language specification adopted by PG carries a great potential in this respect, and allows focusing on the modelling of a large spectrum of linguistic phenomena, especially those involving gradience.

Another important avenue for further work concerns the linguistic substantiation of the different factors that we have postulated as being of some form of influence on graded acceptability judgements, and for which we have already made suggestions.

6.2.3 Generalisation and Optimisation of Loose Constraint-based Parsing

Another avenue of further investigation involves optimising our implementation of LSCP. Conceiving the parsing process as a configuration task, and implementing it as a Constraint Satisfaction process opens a whole range of possibilities regarding its implementation. Configuration tasks are found in very large-scale industrial problems, such as product configuration or software configuration, and require a high level of robustness. Ideally, we would aim to see the parsing activity just as any other large-scale configuration task, so as not to have to worry about algorithmic aspects of constraint satisfaction, and concentrate on problems of language modelling. Different works are already heading in that direction. Haji-Abdolhosseini (2005), for instance, investigates the possibility of applying a semiring-based framework for Constraint Satisfaction Problems (CSPs) (Bistarelli, Montanari, and Rossi, 1997; Bistarelli et al., 1999; Bistarelli and Rossi, 2001) to language processing. The semiring structure proposed by Bistarelli et al. provides an interesting framework for generalising different classes of CSPs, including classical CSP, Valued CSPs, fuzzy CSPs, weighted CSPs, and more. Haji-Abdolhosseini (2005) shows that Keller's Linear Optimality Theory can be seen as an instance of the class of Semiring-based CSPs (SCSPs). Starting from these works, it would be interesting to see how Loose Satisfaction fits into the SCSP framework, and how the different aspects of gradience in natural language that we have discussed could be modelled in such a framework.

Bibliography

Bas Aarts, David Denison, Evelien Keizer, and Gergana Popova., eds. 2004. *Fuzzy Grammar: A Reader*. Cambridge University Press.

Bas Aarts. 2004a. Conceptions of Gradience in the History of Linguistics. *Language Sciences* 26(4):343–389.

Bas Aarts. 2004b. Modelling Linguistic Gradience. *Studies in Language* 28(1):1–49.

Bas Aarts. 2007. *Syntactic gradience: the nature of grammatical indeterminacy.* Oxford University Press.

Bas Aarts. May 2007. In defence of distributional analysis, pace croft. *Studies in Language* 31:431–443(13).

Anne Abeillé, and Marie-Hélène Candito., eds. 2000. *FTAG: A Lexicalized Tree Adjoining Grammar for French, Tree Adjoining Grammars: Formalisms.* CSLI Publications. 305–330.

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. *Building a Treebank for French.* Kluwer Academic Publishers. chapter 10.

Jean-Marie Balfourier, Philippe Blache, Marie-Laure Guénot, and Tristan Vanrullen. 2005. Comparaison de trois analyseurs symboliques pour une tâche d'annotation syntaxique. In *TALN.*

E.G. Bard, D. Robertson, and A. Sorace. 1996. Magnitude Estimation of Linguistic Acceptability. *Language* 72(1):32–68.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Annemarie Walsh, and Timothy Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of InSTIL/ICALL2004–NLP and Speech Technologies in Advanced Language Learning Systems–Venice*, volume 17, 19.

Benjamin K. Bergen, and Nancy Chang. 2005. Embodied Construction Grammar in Simulation-Based Language Understanding. *Construction Grammars: Cognitive Grounding And Theoretical Extensions.*

Gabriel Bès, and Philippe Blache. 1999. Propriétés et analyse d'un langage. In *Proceedings of the 1999 Conference on Traitement Automatique du Langage Naturel (TALN'99).*

Gabriel Bès, Philippe Blache, and Caroline Hagège. 1999. The 5P Paradigm. Technical report, GRIL/LPL.

Joachim De Beule, and Luc Steels. 2005. Hierarchy in Fluid Construction Grammar. In *Proceedings of KI-2005: Advances in Artificial Intelligence*, volume 3698 of *Lecture Notes in Artificial Intelligence*, 1–15. Koblenz, Germany: Springer.

Stefano Bistarelli, and Francesca Rossi. 2001. Semiring-based constraint logic programming: syntax and semantics. *ACM Transactions on Programming Languages and Systems* 23(1):1–29.

Stefano Bistarelli, Ugo Montanari, Francesca Rossi, Thomas Schiex, Gérard Verfaillie, and Héène Fargier. 1999. Semiring-Based CSPs and Valued CSPs: Frameworks, Properties,and Comparison. *Constraints* 4(3):199–240.

Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. 1997. Semiring-Based Constraint Satisfaction and Optimization. *Journal of the ACM* 44(2):201–236.

Philippe Blache, and Jean-Philippe Prost. 2005. Gradience, Constructions and Constraint Systems. In Henning Christiansen, Peter Rossen Skadhauge, and Jorgen Villadsen., eds., *Constraint Solving and Language Processing - CSLP 2004*, volume 3438 of *Lecture Notes in Artificial Intelligence (LNAI).* Roskilde, Denmark: Springer. 74–89. (revised and extended version).

Philippe Blache, and Stéphane Rauzy. 2008. Influence de la qualité de létiquetage sur le chunking: une corrélation dépendant de la taille des chunks. In *Traitement Automatique des Langues Naturelles.*

Philippe Blache, Barbara Hemforth, and Stéphane Rauzy. 2006. Acceptability prediction by means of grammaticality quantification. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 57–64. Sydney, Australia: Association for Computational Linguistics.

Philippe Blache. 2000. Constraints, Linguistic Theories and Natural Language Processing. In D. Christodoulakis., ed., *Natural Language Processing*, volume 1835 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer-Verlag.

Philippe Blache. 2001. *Les Grammaires de Propriétés : des contraintes pour le traitement automatique des langues naturelles.* Hermès Sciences.

Philippe Blache. 2004. Constraints: an operational framework for Constructions Grammars. In *ICCG-04*, 25–26.

Philippe Blache. 2005. Property Grammars: A Fully Constraint-based Theory. In Henning Christiansen, Peter Rossen Skadhauge, and Jorgen Villadsen., eds., *Constraint Solving and Language Processing*, volume 3438 of *LNAI*. Springer.

Dwight Bolinger. 1961. *Generality, Gradience and the All–or–None.* The Hague: Mouton. chapter 1, Gradience. Reprinted in (Aarts et al., 2004, ch. 18).

Denis Bouchard. 1995. *The Semantics of Syntax: A Minimalist Approach to Grammar.* The University of Chicago Press. chapter 1.5.1, Fuzziness and Categorization. Reprinted in (Aarts et al., 2004, ch. 27).

Gosse Bouma. 1992. Feature structures and nonmonotonicity. *Computational Linguistics* 18(2):183–203.

Joan Bresnan, and Tatiana Nikitina. 2003. On the Gradience of the Dative Alternation. Draft.

Jaime G. Carbonell, and Philip J. Hayes. 1983. Recovery strategies for parsing extragrammatical language. *Computational Linguistics* 9(3-4):123–146.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures.* Cambridge University Press.

Noam Chomsky. 1961. *Some methodological remarks on generative grammar.* 219–239. Section 5 reprinted as *Degrees of Grammaticalness* in (Aarts et al., 2004).

Noam Chomsky. 1981. *Lectures on Government and Binding: The Pisa Lectures.* Dordrecht, Foris.

Henning Christiansen. 2005. CHR Grammars. *Theory and Practice of Logic Programming* Special issue on Constraint Handling Rules. [to appear].

William Croft. 2001. *Radical Construction Grammar: Syntactic Theory in Typological Perspective.* Oxford University Press.

William Croft. May 2007. Beyond aristotle and gradience: A reply to aarts. *Studies in Language* 31:409–430(22).

Veronica Dahl, and Philippe Blache. 2004. Directly Executable Constraint Based Grammars. In *Journées Francophones de Programmation en Logique avec Contraintes (JPFLC'2004)*, 149–166.

Ralph Debusmann, Denys Duchier, and Geert-Jan M. Kruijff. 2004. Extensible Dependency Grammar: A New Methodology. In *Proceedings of the 7th International Conference on Computational Linguistics (COLING 2004)*.

Ralph Debusmann, Denys Duchier, and Marco Kuhlmann. 2005. Multi-dimensional Graph Configuration for Natural Language Processing. In Henning Christiansen, Peter Rossen Skadhauge, and Jorgen Villadsen., eds., *Constraint Solving and Language Processing (CSLP 2004), Revised Selected and Invited Papers*, Lecture Notes in Artificial Intelligence (LNAI) 2428. Springer. 104–120.

Edsger W. Dijkstra. 1968. A constructive approach to the problem of program correctness. *BIT Numerical Mathematics* 8(3):174–186.

Shona Douglas, and Robert Dale. 1992. Towards Robust PATR. In *Proceedings of the 14th conference on Computational linguistics*, volume 2, 468–474. Morristown, NJ, USA: Association for Computational Linguistics.

Denys Duchier. 1999. Axiomatizing Dependency Parsing Using Set Parsing. In *Proceedings 6th Meeting on the Mathematics of Language*, 115–126.

Denys Duchier. 2001. Lexicalized Syntax and Topology for Non-projective Dependency Grammar. In *Proceedings of Formal Grammar / Mathematics Of Language (FGMOL-01)*.

Denys Duchier. 2003. Configuration of Labeled Trees under Lexicalized Constraints and Principles. *Research on Language & Computation* 1(3):307–336.

Nigel Duffield. 2003. *The Lexicon Syntax Interface in Second Language Acquisition*. John Benjamins Publishing Company. chapter Measures of Competent Gradience, 97–127.

Mathieu Estratat, and Laurent Henocque. 2004a. Application des programmes de contraintes orientés objet à l'analyse du langage naturel. In *TALN '04*, 163–172.

Mathieu Estratat, and Laurent Henocque. 2004b. Parsing languages with a configurator. In *proceedings of the European Conference for Artificial Intelligence ECAI 2004*, 591–595.

Mathieu Estratat, and Laurent Henocque. 2005a. An Intuitive Tool for Constraint Based Grammars. In Henning Christiansen, Peter Rossen Skadhauge, and Jorgen Villadsen., eds., *Revised Selected and Invited Papers, Constraint Solving and Language Processing, First International Workshop, CSLP 2004, Roskilde, Denmark, September 1–3*, volume 3438 of *Lecture Notes in Computer Science*, 121–139. CSLP. isbn 3–540-26165–6.

Mathieu Estratat, and Laurent Henocque. 2005b. Comprendre des Descriptions Simples à l'aide d'un Configurateur. In *Premières Journées Francophones de Programmation par Contraintes, JFPC'2005*, 325–334.

Mathieu Estratat. 2006. *Vers les grammaires de configuration.* thèse de doctorat, Université Aix-Marseille III. directeur : Laurent Henocque.

Jennifer Foster, and Carl Vogel. 2004. Parsing Ill-Formed Text Using an Error Grammar. *Artificial Intelligence Review* 21(3):269–291.

Kilian Foth, Wolfgang Menzel, and Ingo Schröder. 2005. Robust Parsing with Weighted Constraints. *Natural Language Engineering* 11(1):1–25.

Kilian Foth, Ingo Schröder, and Wolfgang Menzel. 2000. A Transformation-based Parsing Technique With Anytime Properties. In *Proceedings of the International Workshop on parsing Technologies (IWPT-2000)*, 89–100.

Kilian Foth. 2004. Writing Weighted Constraints for Large Dependency Grammars. *Proceedings of Recent Advances in Dependency Grammars, COLING-Workshop.*

Thom Fruehwirth. 1995. Constraint Handling Rules. In A. Podelski., ed., *Constraint Programming: Basics and Trends.* Springer LNCS 910.

Thomas Fruehwirth. 1998. Theory and Practice of Constraint Handling Rules. *The Journal of Logic Programming* 37(1):95–138.

Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar.* Harvard University Press Cambridge.

Véronique Gendner, Gabriel Illouz, Michèle Jardino, Laura Monceaux, Patrick Paroubek, Isabelle Robba, and Anne Vilnat. 2003. PEAS, the first instantiation of a comparative framework for evaluating parsers of French. In *Research Notes of EACL 2003.*

Edward Gibson, Carson T. Schütze, and Ariel Salomon. 1996. The relationship between the frequency and the processing complexity of linguistic structure. *Journal of Psycholinguistic Research* 25(1):59–92.

Edward Gibson. 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition* 68(1):1–76.

Edward Gibson. 2000. The Dependency Locality Theory: A Distance-Based Theory of Linguistic Complexity. In Alec Marantz, Yasushi Miyashita, and Wayne ONeil., eds., *Image, Language, Brain*. Cambridge, Massachussetts, MIT Press. 95–126.

Paul Gochet, and Pascal Gribomont. 1990. *Logique. Méthodes pour linformatique fondamentale*, volume 1.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago University Press.

Danier Grodner, and Edward Gibson. 2005. Consequences of the serial nature of linguistic input for sentential complexity. *Cognitive Science* 29:261–290.

Jeff Gruber, and Edward Gibson. 2004. Measuring linguistic complexity independent of plausibility. *Language* 80(3):583–590.

Marie-Laure Guénot. 2006. *Éléments de grammaire du français pour une théorie descriptive et formelle de la langue*. Spécialité Linguistique, Université de Provence, École Doctorale Cognition, Langage, Education.

Mohammad Haji-Abdolhosseini. 2005. *Modularity And Soft Constraints: A Study of Conflict Resolution in Grammar*. Ph.D. Dissertation, University of Toronto.

Bruce P. Hayes. 2000. *Optimality Theory: Phonology, Syntax, and Acquisition*. Oxford University Press. chapter Gradient Well-Formedness in Optimality Theory.

Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel, and Ingo Shröder. 1998. Eliminative Parsing with Graded Constraints. In *Proceedings 7th International Conferenceon Computational Linguistics, 36th Annual Meeting of the ACL*, volume Coling–ACL '98, pp. 526–530.

Richard Hudson. 1990. *English word grammar*. B. Blackwell Cambridge, Mass., USA.

Dominic Hyde. Fall 2005. Sorites paradox. In Edward N. Zalta., ed., *The Stanford Encyclopedia of Philosophy*.

Ray Jackendoff. 1983. *Semantics and Cognition*. Cambridge, Massachussets, USA: MIT Press. chapter 5, Categorization, Fuzziness, and Family Resemblances. Reprinted in (Aarts et al., 2004, ch. 8).

Daniel Jurafsky, and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR Upper Saddle River, NJ, USA. Errata available at http://www.cs.colorado.edu/∼martin/SLP/slp-errata.html, consulted January 2008.

Robert T. Kasper, and William C. Rounds. 1986. A logical semantics for feature structures. In *Proceedings of the 24th annual meeting on Association for Computational Linguistics*, 257–266. Association for Computational Linguistics Morristown, NJ, USA.

Paul Kay, and Charles Fillmore. 1999. Grammatical Constructions and Linguistic Generalizations: the *what's x doing y?* Construction. *Language* 75(1):1–33.

Paul Kay. 1998. An Informal Sketch of a Formal Architecture for Construction Grammar. In *Conference on Formal Grammar, HPSG and Categorial Grammar.*

Paul Kay. 2002. An Informal Sketch of a Formal Architecture for Construction Grammar. *Grammars* 5(1):1–19.

Frank Keller. 2000. *Gradience in Grammar - Experimental and Computational Aspects of Degrees of Grammaticality.* Ph.D. Dissertation, University of Edinburgh.

Frank Keller. 2003. A Probabilistic Parser as a Model of Global Processing Difficulty. In *Proceedings of ACCSS-03.*

Frank Keller. 2006. Linear optimality theory as a model of gradience in grammar. In Gisbert Fanselow, Caroline Féry, Ralph Vogel, and Matthias Schlesewsky., eds., *Gradience in grammar: Generative perspectives.* New York, USA: Oxford University Press. 270–287.

KilianFoth, Wolfgang Menzel, and Ingo Schröder. 2000. A Transformation-based Parsing Techniques With Anytime Properties. In *Proc. of the International Workshop on Parsing Technologies (IWPT-2000)*, pp 89–100.

George Lakoff. 1973. Fuzzy grammar and the performance/competence terminology game. In *Papers from the ninth regional meeting of the Chicago Linguistic Society.*

George Lakoff. 1977. Linguistic Gestalts. volume 13, 236–287.

George Lakoff. 1987a. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*, In (Lakoff, 1987b). chapter 1, The Importance of Categorization. Reprinted in (Aarts et al., 2004, ch. 10).

George Lakoff. 1987b. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. University of Chicago Press Chicago.

Ronald W. Langacker. 1986. An Introduction to Cognitive Grammar. *Cognitive Science* 10(1):1–40.

Ronald W. Langacker. 1987. *Foundations of Cognitive Grammar*. Stanford University Press.

Alex Lascarides, and Ann Copestake. 1999. Default Representation in Constraint-based Frameworks. *Computational Linguistics* 25(1):55–105.

Daniel Mailharro. 1998. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 12(4):383–397.

Robert Malouf. 2007. Maximal Consistent Subsets. *Computational Linguistics* 33(2):153–160.

Hiroshi Maruyama. 1990. Structural Disambiguation with Constraint Propagation. In *Proceedings 28th Annual Meeting of the ACL*, 31–38.

Kathleen McCoy, Christopher Pennington, and Linda Suri. 1996. English error correction: A syntactic user model based on principled ”mal-rule” scoring. In *Proceedings of UM-96, the Fifth International Conference on User Modeling. Kailua-Kona, Hawaii, January.*, 59–66.

Chris Mellish. 1989. Some Chart-Based Techniques for Parsing Ill-Formed Input. In *Meeting of the Association for Computational Linguistics*, 102–109.

Wolfgang Menzel. 1995. Robust Processing of Natural Language. *Proceedings of the 19th Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence* 19–34.

Wolfgang Menzel. 1998. Constraint satisfaction for robust parsing of spoken language. *Journal of Experimental & Theoretical Artificial Intelligence* 10(1):77–89.

Sanjay Mittal, and Felix Frayman. 1989. Towards a generic model of configuration tasks. In N. S. Sridharan., ed., *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, volume 2, 1395–1401.

Frank Morawietz, and Philippe Blache. 2002. Parsing Natural Languages with CHR. Under consideration for publication in Theory and Practice of Logic Programming.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2006. GLEU: Automatic Evaluation of Sentence-Level Fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 344–351. Association for Computational Linguistics.

Frederick J. Newmeyer. 2000. The Discrete Nature of Syntactic Categories: Against a Prototype-based Account. In Robert Borsley., ed., *Syntax and Semantics*, volume 32: The Nature and Function of Syntactic Categories. Academic Press. 221–250. Reprinted in (Aarts et al., 2004, ch. 28).

Patrick Paroubek, Isabelle Robba, and Christelle Ayache. 2007. Les résultats de la campagne EASY d'évaluation des analyseurs syntaxiques du français. In *Actes de TALN 2007, 14ème conférence sur le Traitement Automatique des Langues Naturelles*, 243–251. Toulouse: IRIT & ERSS.

Patric Paroubek, Isabelle Robba, and Anne Vilnat. 2003. EASY : An Evaluation Protocol for Syntactic Parsers. http://www.limsi.fr/RS2005/chm/lir/lir11/. (as of August 2008).

Carl Pollard, and Ivan Sag. 1994. *Head-driven Phrase Structure Grammars*. Center for the Study of Language and Information Publication (CSLI), Chicago University Press.

Emil L. Post. 1943. Formal Reductions of the General Combinatorial Decision Problem. *American Journal of Mathematics* 65(2):197–215.

Alan Prince, and Paul Smolensky. 1993. Optimality Theory: Constraint Interaction in Generatire Grammar. Technical report, TR-2, Rutgers University Cognitive Science Center, New Brunswick, NJ.

Geoffrey Pullum, and Barbara Scholz. 2001. On the Distinction Between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. In Philippe de Groote, Glyn Morrill, and Christian Rétoré., eds., *Logical Aspects of Computational Linguistics: 4th International Conference*, number 2099 in Lecture Notes in Artificial Intelligence, 17–43. Berlin: Springer Verlag.

Geoffrey K. Pullum, and Barbara C. Scholz. 2005. Contrasting Applications of Logic in Natural Language Syntactic Description. In Petr Hájek, Luis Valdés-Villanueva, and Dag Westerståhl., eds., *Logic, Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress*, 481–503. KCL, The Strand, London WC2R 2LS, U.K.: Department of Computer Science.

Geoffrey K. Pullum. 2007. The Evolution of Model-Theoretic Frameworks in Linguistics. In James Rogers, and Stephan Kepser., eds., *Model-Theoretic Syntax at 10, Proceedings of the MTS@10 workshop, organised as part of ESSLLI 2007, the European Summer School in Logic, Language and Information*, 1–10. Dublin, Ireland: Trinity College Dublin.

Martin Riegel, Jean-Christophe Pellat, and René Rioul. 1998. *Grammaire méthodique du français*. Presses Universitaires de France.

John R. Ross. 1972. The category squish: Endstation Hauptwort. In P. M. Peranteau, J. N. Levi, and G. C. Phares., eds., *Papers from the eighth regional meeting of the Chicago Linguistics Society*, 316–328.

John R. Ross. 1973. Nouniness. *Three Dimensions of Linguistic Research* 137–257. Reprinted in (Aarts et al., 2004, ch. 22).

Daniel Sabin, and Eugene C. Freuder. 1996. Configuration as Composite Constraint Satisfaction. In George F. Luger., ed., *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop*, 153–161. Albuquerque, New Mexico: AAAI press.

Ingo Schröder, Wolfgang Menzel, Kilian Foth, and Michael Schulz. 2000. Modeling Dependency Grammar with Restricted Constraints. *International Journal Traitement Automatique des Langues: Grammaires de dépendance* 41(1):113–142.

Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. Dissertation, Universität Hamburg.

Carson T. Schütze. 1996. *The Empirical Base of Linguistics: Grammaticality Judgments and Linguistic Methodology*. The University of Chicago Press. chapter 3.3, The Nature of Graded Judgments, 62–81. Reprinted in (Aarts et al., 2004, ch. 24).

Stewart Shapiro. Fall 2007. Classical logic. In Edward N. Zalta., ed., *The Stanford Encyclopedia of Philosophy*.

Antonella Sorace, and Frank Keller. 2005. Gradience in linguistic data. *Lingua* 115(11):1497–1524.

Luc Steels, and Joachim de Beule. 2006a. A (Very) Brief Introduction to Fluid Construction Grammar. In *Third International Workshop on Scalable Natural Language Understanding (ScaNaLU 2006)*.

Luc Steels, and Joachim de Beule. 2006b. Unify and Merge in Fluid Construction Grammar. In P. Vogt, Y. Sugita, E. Tuci, and C. Nehaniv., eds., *Symbol Grounding and Beyond. Proceedings of the Third symposium on the Emergence and Evolution of Linguistic Communication, EELC2006*, volume 4211 of *Lecture Notes in Artificial Intelligence*. Rome, Italy: Springer.

Luc Steels, Joachim de Beule, and Nicola Neubauer. 2005. Linking in fluid construction grammars. In *Proceedings of BNAIC, Transactions of the Belgian Royal Society of Arts and Sciences*, 11–18.

Tristan VanRullen, Philippe Blache, Cristel Portes, Sthéphane Rauzy, Jean-François Maeyhieux, Marie-Laure Guénot, Jean-Marie Balfourier, and Emmanuel Bellengier. 2005. Une plateforme pour lacquisition, la maintenance et la validation de ressources lexicales. In *Actes de TALN 2005*.

Tristan VanRullen, Marie-Laure Guénot, and Emmanuel Bellengier. 2003. Formal Representation of Property Grammars. In Balder ten Cate., ed., *Proceedings of Eighth ESSLLI Student Session*.

Tristan VanRullen. 2005. *Vers une analyse syntaxique à granularité variable*. Ph.D. Dissertation, Université de Provence, Informatique.

Anne Vilnat, Patrick Paroubek, Laura Monceaux, Isabelle Robba, Véronique Gendner, Gabriel Illouz, and Michèle Jardino. 2003. EASY or How Difficult Can It Be to Define a Reference Treebank for French. In *2nd Workshop on Treebanks and Linguistic Theories (TLT)*.

Anne Vilnat, Patrick Paroubek, Laura Monceaux, Isabelle Robba, Véronique Gendner, Gabriel Illouz, and Michèle Jardino. 2004. The Ongoing Evaluation Campaign of Syntactic Parsing of French: EASY. In *Proceedings of LREC 2004*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2005. Towards Statistical Paraphrase Generation: Preliminary Evaluations of Grammaticality. In *Proceedings of The 3rd International Workshop on Paraphrasing (IWP2005)*.

T. Warren, and E. Gibson. 2002. The influence of referential processing on sentence complexity. *Cognition* 85(1):79–112.

Peter Van Weert, Tom Schrijvers, and Bart Demoen. 2005. K.U.Leuven JCHR: a user-friendly, flexible and efficient CHR system for Java. In Tom Schrijvers, and Thom Frühwirth., eds., *Proceedings of the Second Workshop on Constraint Handling Rules (CHR 2005)*, 47–62.

Ralph M. Weischedel, and Norman K. Sondheimer. 1983. Meta-rules as a Basis for Processing Ill-formed Input. *American Journal of Computational Linguistics* 9(3-4):161–177.

Eric Weisstein. 2004. k-Subset. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/k-Subset.html.

Eric Weisstein. 2006. Set Partition. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/SetPartition.html.

Index

Appendix A

PG Construction Grammar

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE grammaire SYSTEM "grammaire.dtd">
<?xml-stylesheet type="text/xsl" href="grammaire.xsl"?>
<grammaire comment="J-Ph. Prost (October 07)" label="GP 13" type="MULTEXT">
<!-- *********************************************************************************** -->
<!-- *********************************************************************************** -->
<!-- ****************************   G R O U P E S   ************************************ -->
<!-- *********************************************************************************** -->
<!-- *********************************************************************************** -->
<!-- *********************************************************************************** -->
    <categorie comment="adjectif et syntagme adjectival" label="A">
<!-- *********************************************************************************** -->
        <traits comment="traits de l&apos;adjectif et du syntagme adjectival" label="A_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="qualificatif" label="f"/>
                <valeur comment="indéfini" label="i"/>
                <valeur comment="ordinal" label="o"/>
                <valeur comment="possessif" label="s"/>
                <valeur comment="cardinal" label="k"/>
            </trait>
            <trait comment="pos3" label="type" type="CARACTERE">
                <valeur comment="comparatif" label="c"/>
                <valeur comment="positif" label="p"/>
            </trait>
            <trait comment="pos4" label="genre" type="CARACTERE">
                <valeur comment="masculin" label="m"/>
                <valeur comment="féminin" label="f"/>
            </trait>
            <trait comment="pos5" label="nombre" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="type"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
        </traits>
    </categorie>
<!-- *********************************************************************************** -->
    <categorie comment="conjonction" label="C">
<!-- *********************************************************************************** -->
        <traits comment="traits de la conjonction" label="C_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="coordination" label="c"/>
                <valeur comment="subordination" label="s"/>
            </trait>
            <propagation comment="pas de trait propagé puisque pas de syntagme" label=""/>
        </traits>
    </categorie>
<!-- *********************************************************************************** -->
    <categorie comment="déterminant" label="D">
<!-- *********************************************************************************** -->
        <traits comment="traits du déterminant" label="D_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="article?" label="a"/>
                <valeur comment="démonstratif" label="d"/>
                <valeur comment="indéfini" label="i"/>
                <valeur comment="possessif" label="s"/>
                <valeur comment="interrogatif" label="t"/>
                <valeur comment="cardinal" label="k"/>
            </trait>
            <trait comment="pos3" label="ordre" type="CARACTERE">
                <valeur comment="première personne" label="1"/>
                <valeur comment="deuxième personne" label="2"/>
                <valeur comment="troisième personne" label="3"/>
            </trait>
            <trait comment="pos4" label="genre" type="CARACTERE">
                <valeur comment="masculin" label="m"/>
                <valeur comment="féminin" label="f"/>
            </trait>
            <trait comment="pos5" label="nombre" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <trait comment="pos6" label="possesseur" type="CARACTERE">
```

```
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <trait comment="pos7" label="resultatFusion" type="CARACTERE">
                <valeur comment="défini" label="d"/>
                <valeur comment="indéfini" label="i"/>
            </trait>
            <trait comment="pos8" label="fusionAvecQuoi" type="CARACTERE">
                <valeur comment="à + le" label="a"/>
                <valeur comment="de + le" label="d"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="ordre"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="possesseur"/>
            <propagation label="resultatFusion"/>
            <propagation label="fusionAvecQuoi"/>
        </traits>
    </categorie>
<!-- *********************************************************************************** -->
    <categorie comment="Nom" label="N">
<!-- *********************************************************************************** -->
        <traits comment="traits du nom" label="N_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="commun" label="c"/>
                <valeur comment="propre avec déterminant" label="d"/>
                <valeur comment="propre sans déterminant" label="p"/>
                <valeur comment="latin?" label="l"/>
                <valeur comment="cardinal" label="k"/>
            </trait>
            <trait comment="pos3" label="genre" type="CARACTERE">
                <valeur comment="masculin" label="m"/>
                <valeur comment="féminin" label="f"/>
            </trait>
            <trait comment="pos4" label="nombre" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <trait comment="pos5" label="sigle" type="CARACTERE">
                <valeur comment="sigle (abréviation, etc.)" label="s"/>
            </trait>
            <trait comment="pos6" label="typeNomPropre" type="CARACTERE">
                <valeur comment="pays" label="c"/>
                <valeur comment="habitants" label="h"/>
                <valeur comment="société" label="s"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="sigle"/>
            <propagation label="typeNomPropre"/>
        </traits>
    </categorie>
<!-- *********************************************************************************** -->
    <categorie comment="pronom" label="P">
<!-- *********************************************************************************** -->
        <traits comment="traits du pronom" label="P_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="démonstratif" label="d"/>
                <valeur comment="indéfini" label="i"/>
                <valeur comment="personnel" label="p"/>
                <valeur comment="relatif" label="r"/>
                <valeur comment="possessif" label="s"/>
                <valeur comment="interrogatif" label="t"/>
                <valeur comment="réfléchi" label="x"/>
                <valeur comment="cardinal" label="k"/>
            </trait>
            <trait comment="pos3" label="ordre" type="CARACTERE">
                <valeur comment="première personne" label="1"/>
                <valeur comment="deuxième personne" label="2"/>
                <valeur comment="troisième personne" label="3"/>
            </trait>
            <trait comment="pos4" label="genre" type="CARACTERE">
                <valeur comment="masculin" label="m"/>
                <valeur comment="féminin" label="f"/>
            </trait>
            <trait comment="pos5" label="nombre" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <trait comment="pos6" label="typePronom" type="CARACTERE">
                <valeur comment="nominatif" label="n"/>
                <valeur comment="accusatif (cod)" label="a"/>
                <valeur comment="datif (coi)" label="d"/>
                <valeur comment="oblique (le reste)" label="o"/>
            </trait>
            <trait comment="pos7" label="possesseur" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="ordre"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="typePronom"/>
            <propagation label="possesseur"/>
        </traits>
    </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="adverbe / syntagme adverbial / syntagme prépositionnel" label="R">
<!-- ****************************************************************************************** -->
        <traits comment="traits de l&apos;adverbe" label="R_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="général" label="g"/>
                <valeur comment="particule" label="p"/>
            </trait>
            <trait comment="pos3" label="type" type="CARACTERE">
                <valeur comment="comparatif" label="c"/>
                <valeur comment="positif" label="p"/>
                <valeur comment="associé à négation" label="d"/>
                <valeur comment="négatif" label="n"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="type"/>
        </traits>
    </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="préposition" label="S">
<!-- ****************************************************************************************** -->
        <traits comment="traits de la préposition" label="S_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="préposition" label="p"/>
            </trait>
            <trait comment="pos3" label="type" type="CARACTERE">
                <valeur comment="à" label="a"/>
                <valeur comment="de" label="d"/>
            </trait>
            <trait comment="pos4" label="composeur" type="CARACTERE">
                <valeur comment="composition" label="+"/>
                <valeur comment="pas de composition" label="0"/>
            </trait>
            <trait comment="pos5" label="categComposeur" type="CARACTERE">
                <valeur comment="Déterminant" label="D"/>
                <valeur comment="Adjectif" label="A"/>
                <valeur comment="Pronom" label="P"/>
                <valeur comment="pas de composition" label="0"/>
            </trait>
            <trait comment="pos6" label="soucatComposeur" type="CARACTERE">
                <valeur comment="Déterminant article?" label="a"/>
                <valeur comment="Déterminant ou Pronom démonstratif" label="d"/>
                <valeur comment="Déterminant ou Adjectif ou Pronom indéfini" label="i"/>
                <valeur comment="Déterminant ou Adjectif ou Pronom possessif" label="s"/>
                <valeur comment="Déterminant ou Pronom interrogatif" label="t"/>
                <valeur comment="Déterminant ou Adjectif ou Pronom cardinal" label="k"/>
                <valeur comment="Adjectif qualificatif" label="f"/>
                <valeur comment="Adjectif ordinal" label="o"/>
                <valeur comment="Pronom personnel" label="p"/>
                <valeur comment="Pronom relatif" label="r"/>
                <valeur comment="Pronom réfléchi" label="x"/>
```

```
                <valeur comment="pas de composition" label="0"/>
            </trait>
            <trait comment="pos7" label="typeComposeur" type="CARACTERE">
                <valeur comment="Adjectif comparatif" label="c"/>
                <valeur comment="Adjectif positif" label="p"/>
                <valeur comment="Déterminant ou Pronom première personne" label="1"/>
                <valeur comment="Déterminant ou Pronom deuxième personne" label="2"/>
                <valeur comment="Déterminant ou Pronom troisième personne" label="3"/>
                <valeur comment="pas de composition" label="0"/>
            </trait>
            <trait comment="pos8" label="genre" type="CARACTERE">
                <valeur comment="masculin" label="m"/>
                <valeur comment="féminin" label="f"/>
                <valeur comment="pas de composition" label="0"/>
            </trait>
            <trait comment="pos9" label="nombre" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
                <valeur comment="pas de composition" label="0"/>
            </trait>
        </traits>
    </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="verbe" label="V">
<!-- ****************************************************************************************** -->
        <traits comment="traits du verbe" label="V_traits">
            <trait comment="pos2" label="soucat" type="CARACTERE">
                <valeur comment="principal" label="m"/>
                <valeur comment="modal" label="o"/>
                <valeur comment="auxiliaire avoir" label="a"/>
                <valeur comment="auxiliaire être" label="e"/>
            </trait>
            <trait comment="pos3" label="mode" type="CARACTERE">
                <valeur comment="infinitif" label="n"/>
                <valeur comment="indicatif" label="i"/>
                <valeur comment="impératif" label="m"/>
                <valeur comment="conditionnel" label="c"/>
                <valeur comment="subjonctif" label="s"/>
                <valeur comment="participe" label="p"/>
            </trait>
            <trait comment="pos4" label="temps" type="CARACTERE">
                <valeur comment="présent" label="p"/>
                <valeur comment="passé" label="s"/>
                <valeur comment="imparfait" label="i"/>
                <valeur comment="futur" label="f"/>
            </trait>
            <trait comment="pos5" label="ordre" type="CARACTERE">
                <valeur comment="première personne" label="1"/>
                <valeur comment="deuxième personne" label="2"/>
                <valeur comment="troisième personne" label="3"/>
            </trait>
            <trait comment="pos6" label="nombre" type="CARACTERE">
                <valeur comment="singulier" label="s"/>
                <valeur comment="pluriel" label="p"/>
            </trait>
            <trait comment="pos7" label="genre" type="CARACTERE">
                <valeur comment="masculin" label="m"/>
                <valeur comment="féminin" label="f"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="mode"/>
            <propagation label="temps"/>
            <propagation label="ordre"/>
            <propagation label="nombre"/>
            <propagation label="genre"/>
        </traits>
    </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="Noyau Verbal" label="NV">
<!-- ****************************************************************************************** -->
        <proprietes comment="0.8" label="proprietes">
            <propriete label="obligation">
                <clause weight="1">
                    <membre>
                        <refCateg comment="verbe" label="V"/>
        </membre>
    </clause>
</propriete>
<propriete label="unicite">
    <clause weight="1">
        <membre>
            <refCateg label="V">
                <refValeur label="soucat" operateur="=" valeur="m"/>
                <refValeur label="mode" operateur="=" valeur="p"/>
                <refValeur label="temps" operateur="=" valeur="s"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="GN"/>
        </membre>
    </clause>
</propriete>
<propriete label="exigence">
    <clause weight="1">
        <membre>
            <refCateg label="V">
                <refValeur label="mode" operateur="=" valeur="p"/>
                <refValeur label="temps" operateur="=" valeur="s"/>
            </refCateg>
        </membre>
        <membre>
            <operationLogique label="OU">
                <refCateg label="V">
                    <refValeur label="soucat" operateur="=" valeur="a"/>
                </refCateg>
                <refCateg label="V">
                    <refValeur label="soucat" operateur="=" valeur="e"/>
                </refCateg>
            </operationLogique>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="V">
                <refValeur label="soucat" operateur="=" valeur="o"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg label="V">
                <refValeur label="mode" operateur="=" valeur="n"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="V">
                <refValeur label="soucat" operateur="=" valeur="a"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg label="V">
                <refValeur label="mode" operateur="=" valeur="p"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="V">
                <refValeur label="soucat" operateur="=" valeur="e"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg label="V">
                <refValeur label="mode" operateur="=" valeur="p"/>
            </refCateg>
        </membre>
    </clause>
</propriete>
```

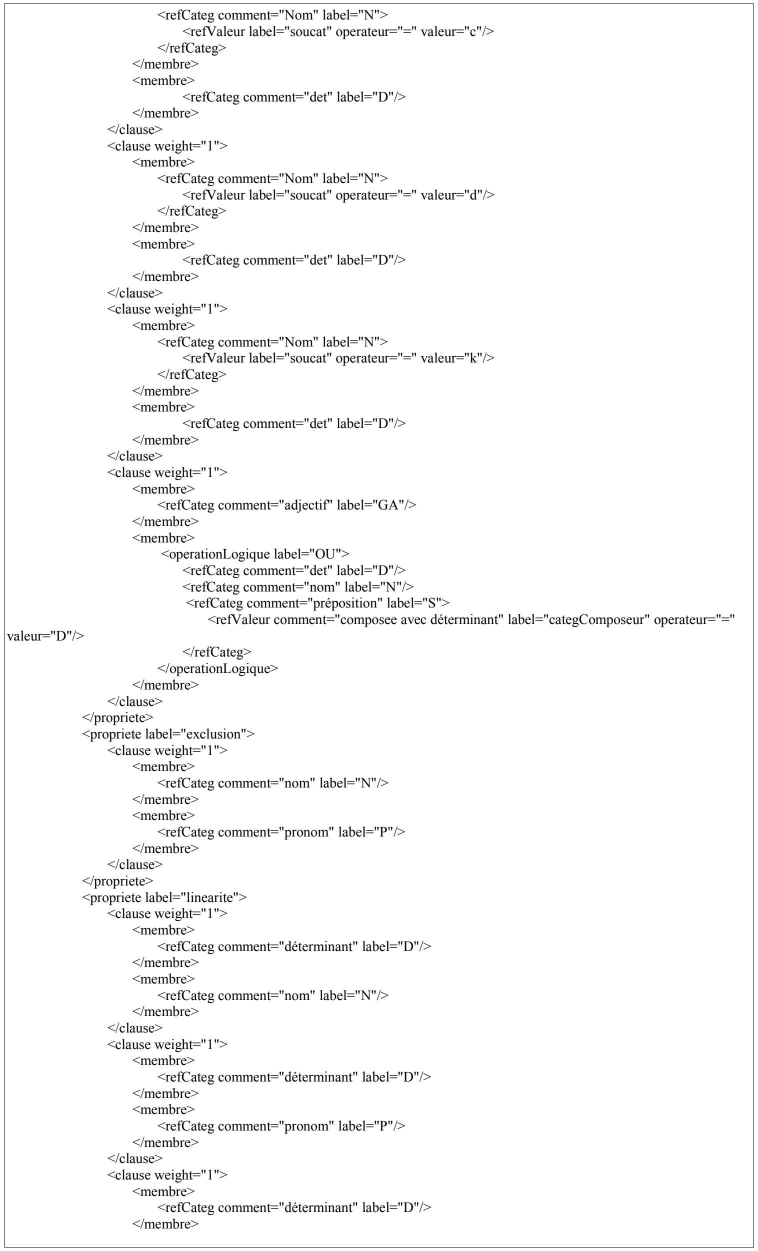
```
<propriete label="exclusion">
    <clause weight="1">
        <membre>
            <refCateg label="P">
                <refValeur label="typePronom" operateur="=" valeur="a"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg label="GN"/>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="P">
                <refValeur label="typePronom" operateur="=" valeur="d"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg label="P">
                <refValeur label="typePronom" operateur="=" valeur="a"/>
            </refCateg>
        </membre>
    </clause>
</propriete>
<propriete label="linearite">
    <clause weight="1">
        <membre>
            <refCateg label="V"/>
        </membre>
        <membre>
            <refCateg label="GN"/>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="V"/>
        </membre>
        <membre>
            <refCateg label="R"/>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="V"/>
        </membre>
        <membre>
            <refCateg label="GP"/>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg label="V"/>
        </membre>
        <membre>
            <refCateg label="Compl"/>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="adverbe" label="R"/>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="accusatif" label="typePronom" operateur="=" valeur="a"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="datif" label="typePronom" operateur="=" valeur="d"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg comment="adverbe" label="R"/>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="accusatif" label="typePronom" operateur="=" valeur="a"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg comment="adverbe" label="R"/>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="datif" label="typePronom" operateur="=" valeur="d"/>
            </refCateg>
        </membre>
    </clause>
    <clause weight="1">
        <membre>
            <refCateg comment="adverbe" label="R"/>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            </refCateg>
        </membre>
    </clause>
</propriete>
<propriete comment="modif TV: il manque les OU" label="dependance">
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="i"/>
                <refTrait label="ordre"/>
```

```
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="m"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="c"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="s"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="n"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
        <membre>
            <refCateg comment="pronom" label="P">
                <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
                <refTrait label="ordre"/>
            </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
        <membre>
            <refCateg comment="verbe" label="V">
                <refValeur comment="pas un participe" label="mode" operateur="=" valeur="i"/>
                <refTrait label="ordre"/>
            </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="m"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="c"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="s"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            <refTrait label="ordre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
            <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="i"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
```

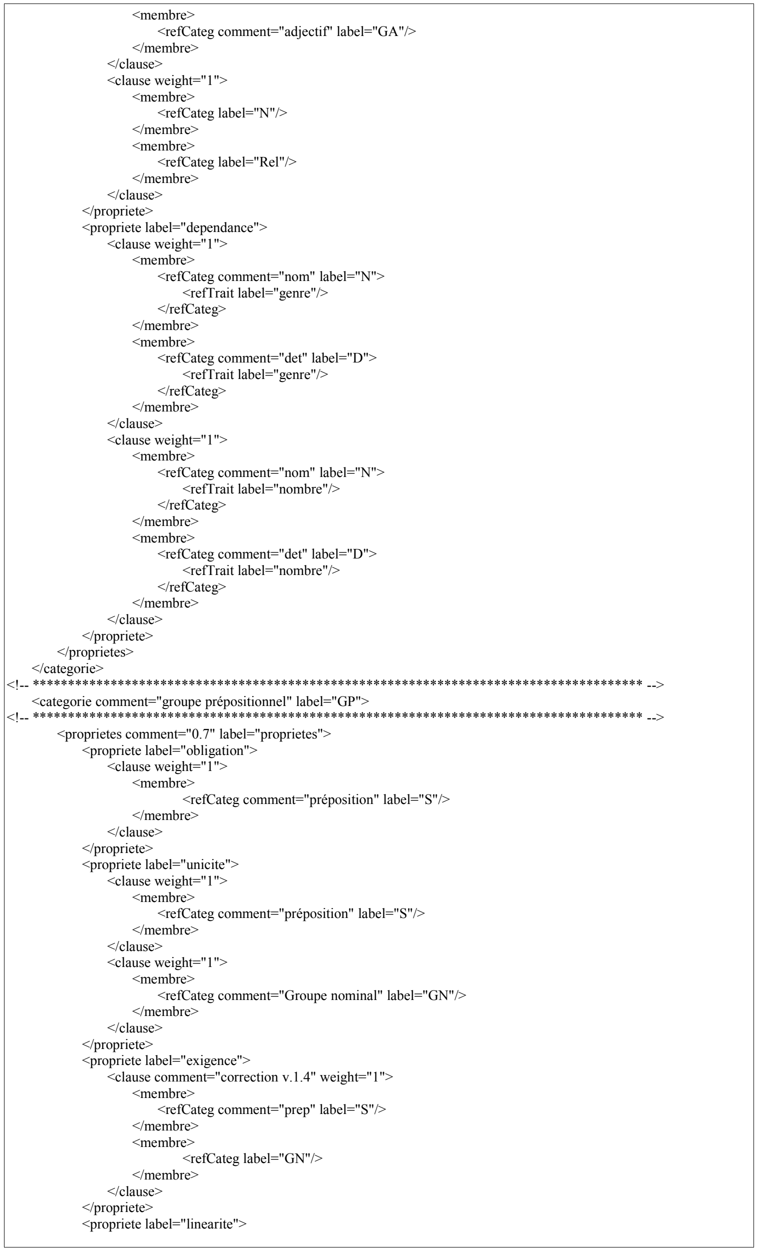
```
            <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="m"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
            <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="c"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
            <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="s"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
            <refValeur comment="nominatif" label="typePronom" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="n"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
    <membre>
        <refCateg comment="verbe" label="V">
            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="i"/>
            <refTrait label="nombre"/>
        </refCateg>
    </membre>
    <membre>
        <refCateg comment="pronom" label="P">
            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
            <refTrait label="nombre"/>
```

```
                        </refCateg>
                    </membre>
                </clause>
                <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
                    <membre>
                        <refCateg comment="verbe" label="V">
                            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="m"/>
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
                    <membre>
                        <refCateg comment="verbe" label="V">
                            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="c"/>
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause comment="ajout TV: les verbes au participe ne sont pas concernés" weight="1">
                    <membre>
                        <refCateg comment="verbe" label="V">
                            <refValeur comment="pas un participe" label="mode" operateur="=" valeur="s"/>
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="réfléchi" label="soucat" operateur="=" valeur="x"/>
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>
<!-- ************************************************************************************ -->
    <categorie comment="groupe nominal" label="GN">
<!-- ************************************************************************************ -->
        <traits comment="traits du groupe nominal" label="GN_traits"/>
        <proprietes comment="0.8" label="proprietes">
            <propriete label="obligation">
                <clause weight="1">
                    <membre>
                        <operationLogique label="OU">
                            <refCateg comment="nom" label="N"/>
                            <refCateg comment="pronom" label="P">
                                <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                            </refCateg>
                            <refCateg comment="pronom" label="P">
                                <refValeur comment="démonstratif" label="soucat" operateur="=" valeur="d"/>
                            </refCateg>
                            <refCateg comment="pronom" label="P">
                                <refValeur comment="indéfini" label="soucat" operateur="=" valeur="i"/>
                            </refCateg>
                            <refCateg comment="pronom" label="P">
                                <refValeur comment="relatif" label="soucat" operateur="=" valeur="r"/>
                            </refCateg>
                            <refCateg comment="pronom" label="P">
                                <refValeur comment="possessif" label="soucat" operateur="=" valeur="s"/>
                            </refCateg>
                            <refCateg comment="pronom" label="P">
                                <refValeur comment="interrogatif" label="soucat" operateur="=" valeur="t"/>
```

```
                            </refCateg>
                            <refCateg comment="adjectif" label="GA"/>
                            <refCateg label="Compl"/>
                        </operationLogique>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">
                <clause weight="1">
                    <membre>
                        <refCateg comment="Nom" label="N"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="Det" label="D"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="Compl"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="Rel"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="personnel" label="soucat" operateur="=" valeur="p"/>
                            <refValeur comment="oblique" label="typePronom" operateur="=" valeur="o"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="démonstratif" label="soucat" operateur="=" valeur="d"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="indéfini" label="soucat" operateur="=" valeur="i"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="relatif" label="soucat" operateur="=" valeur="r"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="possessif" label="soucat" operateur="=" valeur="s"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="pronom" label="P">
                            <refValeur comment="interrogatif" label="soucat" operateur="=" valeur="t"/>
                        </refCateg>
                    </membre>
                </clause>
            </propriete>
            <propriete label="exigence">
                <clause weight="1">
                    <membre>
```



```
                        <refCateg comment="Nom" label="N">
                            <refValeur label="soucat" operateur="=" valeur="c"/>
                        </refCateg>
                    </membre>
                    <membre>
                            <refCateg comment="det" label="D"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="Nom" label="N">
                            <refValeur label="soucat" operateur="=" valeur="d"/>
                        </refCateg>
                    </membre>
                    <membre>
                            <refCateg comment="det" label="D"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="Nom" label="N">
                            <refValeur label="soucat" operateur="=" valeur="k"/>
                        </refCateg>
                    </membre>
                    <membre>
                            <refCateg comment="det" label="D"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="adjectif" label="GA"/>
                    </membre>
                    <membre>
                        <operationLogique label="OU">
                            <refCateg comment="det" label="D"/>
                            <refCateg comment="nom" label="N"/>
                            <refCateg comment="préposition" label="S">
                                <refValeur comment="composee avec déterminant" label="categComposeur" operateur="="
valeur="D"/>
                            </refCateg>
                        </operationLogique>
                    </membre>
                </clause>
            </propriete>
            <propriete label="exclusion">
                <clause weight="1">
                    <membre>
                        <refCateg comment="nom" label="N"/>
                    </membre>
                    <membre>
                        <refCateg comment="pronom" label="P"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="linearite">
                <clause weight="1">
                    <membre>
                        <refCateg comment="déterminant" label="D"/>
                    </membre>
                    <membre>
                        <refCateg comment="nom" label="N"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="déterminant" label="D"/>
                    </membre>
                    <membre>
                        <refCateg comment="pronom" label="P"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="déterminant" label="D"/>
                    </membre>
```



```
                    <membre>
                        <refCateg comment="adjectif" label="GA"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="N"/>
                    </membre>
                    <membre>
                        <refCateg label="Rel"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="dependance">
                <clause weight="1">
                    <membre>
                        <refCateg comment="nom" label="N">
                            <refTrait label="genre"/>
                        </refCateg>
                    </membre>
                    <membre>
                        <refCateg comment="det" label="D">
                            <refTrait label="genre"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="nom" label="N">
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                    <membre>
                        <refCateg comment="det" label="D">
                            <refTrait label="nombre"/>
                        </refCateg>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="groupe prépositionnel" label="GP">
<!-- ****************************************************************************************** -->
        <proprietes comment="0.7" label="proprietes">
            <propriete label="obligation">
                <clause weight="1">
                    <membre>
                            <refCateg comment="préposition" label="S"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">
                <clause weight="1">
                    <membre>
                        <refCateg comment="préposition" label="S"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg comment="Groupe nominal" label="GN"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="exigence">
                <clause comment="correction v.1.4" weight="1">
                    <membre>
                        <refCateg comment="prep" label="S"/>
                    </membre>
                    <membre>
                            <refCateg label="GN"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="linearite">
        <clause weight="1">
          <membre>
            <refCateg comment="préposition" label="S"/>
          </membre>
          <membre>
            <refCateg comment="groupe nominal" label="GN"/>
          </membre>
        </clause>
        <clause weight="1">
          <membre>
            <refCateg comment="préposition" label="S"/>
          </membre>
          <membre>
            <refCateg comment="noyau verbal" label="NV"/>
          </membre>
        </clause>
      </propriete>
      <propriete label="dependance">
        <clause weight="1">
          <membre>
            <refCateg label="GN"/>
          </membre>
          <membre>
            <refCateg label="S"/>
          </membre>
        </clause>
      </propriete>
    </proprietes>
  </categorie>
<!-- ***************************************************************************************** -->
  <categorie comment="groupe adjectival" label="GA">
<!-- ***************************************************************************************** -->
    <proprietes comment="0.5" label="proprietes">
      <propriete label="obligation">
        <clause weight="1">
          <membre>
            <operationLogique label="OU">
            <refCateg comment="adjectif" label="A"/>
            <refCateg comment="verbe" label="V">
              <refValeur comment="participe" label="mode" operateur="=" valeur="p"/>
              <refValeur comment="temps" label="temps" operateur="=" valeur="s"/>
            </refCateg>
            </operationLogique>
          </membre>
        </clause>
      </propriete>
      <propriete label="unicite">
        <clause weight="1">
          <membre>
            <refCateg comment="adjectif" label="A"/>
          </membre>
        </clause>
        <clause weight="1">
          <membre>
            <refCateg comment="verbe" label="V">
              <refValeur comment="participe" label="mode" operateur="=" valeur="p"/>
              <refValeur comment="temps" label="temps" operateur="=" valeur="s"/>
            </refCateg>
          </membre>
        </clause>
        <clause weight="1">
          <membre>
            <refCateg comment="adverbe" label="R"/>
          </membre>
        </clause>
      </propriete>
      <propriete label="exclusion">
        <clause weight="1">
          <membre>
            <refCateg comment="adjectif" label="A"/>
          </membre>
          <membre>
            <refCateg comment="verbe" label="V">
              <refValeur comment="participe" label="mode" operateur="=" valeur="p"/>
              <refValeur comment="temps" label="temps" operateur="=" valeur="s"/>
```

```
            </refCateg>
          </membre>
        </clause>
      </propriete>
      <propriete label="linearite">
        <clause weight="1">
          <membre>
            <refCateg comment="adverbe" label="R"/>
          </membre>
          <membre>
            <refCateg comment="adjectif" label="A"/>
          </membre>
        </clause>
      </propriete>
    </proprietes>
  </categorie>
<!-- ***************************************************************************************** -->
  <categorie comment="Proposition relative" label="Rel">
<!-- ***************************************************************************************** -->
    <traits label="Pas de traits pour la categorie Relative"/>
    <proprietes comment="0.9" label="proprietes">
      <propriete label="obligation">
        <clause weight="1">
          <membre>
            <refCateg comment="pronom" label="P">
              <refValeur label="soucat" operateur="=" valeur="r"/>
            </refCateg>
          </membre>
        </clause>
      </propriete>
      <propriete label="unicite">
        <clause weight="1">
          <membre>
            <refCateg comment="pronom" label="P">
              <refValeur label="soucat" operateur="=" valeur="r"/>
            </refCateg>
          </membre>
        </clause>
        <clause weight="1">
          <membre>
            <refCateg comment="phrase" label="S"/>
          </membre>
        </clause>
        <clause weight="1">
          <membre>
            <refCateg comment="noyau verbal" label="NV"/>
          </membre>
        </clause>
      </propriete>
      <propriete label="exclusion">
        <clause weight="1">
          <membre>
            <refCateg comment="phrase" label="S"/>
          </membre>
          <membre>
            <refCateg comment="Noyau verbal" label="NV"/>
          </membre>
        </clause>
      </propriete>
      <propriete label="linearite">
        <clause weight="1">
          <membre>
            <refCateg label="P">
              <refValeur label="soucat" operateur="=" valeur="r"/>
            </refCateg>
          </membre>
          <membre>
            <refCateg label="Ut"/>
          </membre>
        </clause>
        <clause weight="1">
          <membre>
            <refCateg label="P">
              <refValeur label="soucat" operateur="=" valeur="r"/>
            </refCateg>
```

```
                    </membre>
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="exigence">
                <clause weight="1">
                    <membre>
                        <refCateg label="Ut"/>
                    </membre>
                    <membre>
                        <refCateg label="P">
                            <refValeur label="soucat" operateur="=" valeur="r"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                    <membre>
                        <refCateg label="P">
                            <refValeur label="soucat" operateur="=" valeur="r"/>
                        </refCateg>
                    </membre>
                </clause>
            </propriete>
            <propriete label="dependance">
                <clause weight="1">
                    <membre>
                        <refCateg label="Ut"/>
                    </membre>
                    <membre>
                        <refCateg label="P">
                            <refValeur label="soucat" operateur="=" valeur="r"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                    <membre>
                        <refCateg label="P">
                            <refValeur label="soucat" operateur="r" valeur="="/>
                        </refCateg>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>
<!-- ****************************************************************************************** -->
 <categorie comment="Coordination Nominale" label="CN">
<!-- ****************************************************************************************** -->
  <traits comment="Traits du CN" label="S_Traits"/>
  <proprietes comment="1" label="Proprietes">
   <propriete label="obligation">
    <clause weight="1">
     <membre>
      <refCateg label="C"/>
     </membre>
    </clause>
   </propriete>
   <propriete label="exigence">
    <clause weight="1">
     <membre>
      <refCateg label="N"/>
     </membre>
     <membre>
      <refCateg label="N"/>
     </membre>
    </clause>
    <clause weight="1">
     <membre>
```

```
      <refCateg label="GN"/>
     </membre>
     <membre>
      <refCateg label="GN"/>
     </membre>
    </clause>
   </propriete>
  </proprietes>
 </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="Enoncé" label="Ut">
<!-- ****************************************************************************************** -->
        <traits comment="Traits de la phrase" label="S_Traits"/>
        <proprietes comment="0.7" label="Proprietes">
            <propriete label="obligation">
                <clause weight="1">
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">
                <clause weight="1">
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="GN"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="linearite">
    <clause weight="1">
     <membre>
      <refCateg label="CN"/>
     </membre>
     <membre>
      <refCateg label="NV"/>
     </membre>
    </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="GN"/>
                    </membre>
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="dependance">
                <clause weight="1">
                    <membre>
                        <refCateg label="GN"/>
                    </membre>
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>
<!-- ****************************************************************************************** -->
    <categorie comment="" label="Compl">
<!-- ****************************************************************************************** -->
        <traits label="Compl_traits"/>
        <proprietes comment="0.9" label="Proprietes">
            <propriete label="obligation">
                <clause weight="1">
                    <membre>
                        <operationLogique label="OU">
                            <refCateg label="C">
                                <refValeur label="soucat" operateur="=" valeur="s"/>
                            </refCateg>
                            <refCateg label="S">
                                <refValeur label="type" operateur="=" valeur="d"/>
                            </refCateg>
                        </operationLogique>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">
                <clause weight="1">
                    <membre>
                        <refCateg label="C">
                            <refValeur label="soucat" operateur="=" valeur="s"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="S">
                            <refValeur label="type" operateur="=" valeur="d"/>
                        </refCateg>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                </clause>
                <clause weight="1">
                    <membre>
                        <refCateg label="Ut"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="exclusion">
                <clause weight="1">
                    <membre>
                        <refCateg label="NV"/>
                    </membre>
                    <membre>
                        <refCateg label="Ut"/>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>
</grammaire>
```

Appendix B

PG EASY Grammar

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE grammaire SYSTEM "grammaire.dtd">
<?xml-stylesheet type="text/xsl" href="grammaire.xsl"?>
<grammaire label="Grammaire de Propriétés EASY" comment="J-Ph. Prost (March 07) ; ML Guénot, T VanRullen | 14-09-2004 | version
2" type="MULTEXT">

<!-- ********************************************************************************************* -->
<!-- ********************************************************************************************* -->
<!-- ******************************     G R O U P E S    ***************************************** -->
<!-- ********************************************************************************************* -->
<!-- ********************************************************************************************* -->

<!-- ********************************************************************************************* -->
    <categorie label="A" comment="adjectif et syntagme adjectival">
<!-- ********************************************************************************************* -->

        <traits label="A_traits" comment="traits de l'adjectif et du syntagme adjectival">
            <trait label="soucat" comment="pos2">
                <valeur label="f" comment="qualificatif"/>
                <valeur label="i" comment="indéfini"/>
                <valeur label="o" comment="ordinal"/>
                <valeur label="s" comment="possessif"/>
                <valeur label="k" comment="cardinal"/>
            </trait>
            <trait label="type" comment="pos3">
                <valeur label="c" comment="comparatif"/>
                <valeur label="p" comment="positif"/>
            </trait>
            <trait label="genre" comment="pos4">
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
            </trait>
            <trait label="nombre" comment="pos5">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="type"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
        </traits>
    </categorie>

<!-- ********************************************************************************************* -->
    <categorie label="C" comment="conjonction">
<!-- ********************************************************************************************* -->

        <traits label="C_traits" comment="traits de la conjonction">
            <trait label="soucat" comment="pos2">
                <valeur label="c" comment="coordination"/>
                <valeur label="s" comment="subordination"/>
            </trait>
            <propagation label="" comment="pas de trait propagé puisque pas de syntagme"/>
        </traits>
    </categorie>

<!-- ********************************************************************************************* -->
    <categorie label="D" comment="déterminant">
<!-- ********************************************************************************************* -->

        <traits label="D_traits" comment="traits du déterminant">
            <trait label="soucat" comment="pos2">
                <valeur label="a" comment="article?"/>
                <valeur label="d" comment="démonstratif"/>
                <valeur label="i" comment="indéfini"/>
                <valeur label="s" comment="possessif"/>
                <valeur label="t" comment="interrogatif"/>
                <valeur label="k" comment="cardinal"/>
            </trait>
            <trait label="ordre" comment="pos3">
                <valeur label="1" comment="première personne"/>
                <valeur label="2" comment="deuxième personne"/>
                <valeur label="3" comment="troisième personne"/>
            </trait>
            <trait label="genre" comment="pos4">
```

```
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
            </trait>
            <trait label="nombre" comment="pos5">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <trait label="possesseur" comment="pos6">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <trait label="resultatFusion" comment="pos7">
                <valeur label="d" comment="défini"/>
                <valeur label="i" comment="indéfini"/>
            </trait>
            <trait label="fusionAvecQuoi" comment="pos8">
                <valeur label="a" comment="à + le"/>
                <valeur label="d" comment="de + le"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="ordre"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="possesseur"/>
            <propagation label="resultatFusion"/>
            <propagation label="fusionAvecQuoi"/>
        </traits>
    </categorie>

<!-- ********************************************************************************************* -->
    <categorie label="N" comment="Nom">
<!-- ********************************************************************************************* -->

        <traits label="N_traits" comment="traits du nom">
            <trait label="soucat" comment="pos2">
                <valeur label="c" comment="commun"/>
                <valeur label="d" comment="propre avec déterminant"/>
                <valeur label="p" comment="propre sans déterminant"/>
                <valeur label="l" comment="latin?"/>
                <valeur label="k" comment="cardinal"/>
            </trait>
            <trait label="genre" comment="pos3">
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
            </trait>
            <trait label="nombre" comment="pos4">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <trait label="sigle" comment="pos5">
                <valeur label="s" comment="sigle (abréviation, etc.)"/>
            </trait>
            <trait label="typeNomPropre" comment="pos6">
                <valeur label="c" comment="pays"/>
                <valeur label="h" comment="habitants"/>
                <valeur label="s" comment="société"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="sigle"/>
            <propagation label="typeNomPropre"/>
        </traits>
    </categorie>

<!-- ********************************************************************************************* -->
    <categorie label="P" comment="pronom">
<!-- ********************************************************************************************* -->

        <traits label="P_traits" comment="traits du pronom">
            <trait label="soucat" comment="pos2">
                <valeur label="d" comment="démonstratif"/>
                <valeur label="i" comment="indéfini"/>
                <valeur label="p" comment="personnel"/>
                <valeur label="r" comment="relatif"/>
                <valeur label="s" comment="possessif"/>
                <valeur label="t" comment="interrogatif"/>
                <valeur label="x" comment="réfléchi"/>
                <valeur label="k" comment="cardinal"/>
            </trait>
            <trait label="ordre" comment="pos3">
                <valeur label="1" comment="première personne"/>
                <valeur label="2" comment="deuxième personne"/>
                <valeur label="3" comment="troisième personne"/>
            </trait>
            <trait label="genre" comment="pos4">
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
            </trait>
            <trait label="nombre" comment="pos5">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <trait label="typePronom" comment="pos6">
                <valeur label="n" comment="nominatif"/>
                <valeur label="a" comment="accusatif (cod)"/>
                <valeur label="d" comment="datif (coi)"/>
                <valeur label="o" comment="oblique (le reste)"/>
            </trait>
            <trait label="possesseur" comment="pos7">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="ordre"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="typePronom"/>
            <propagation label="possesseur"/>
        </traits>
    </categorie>

<!-- ****************************************************************************************** -->
    <categorie label="R" comment="adverbe / syntagme adverbial / syntagme prépositionnel">
<!-- ****************************************************************************************** -->

        <traits label="R_traits" comment="traits de l'adverbe">
            <trait label="soucat" comment="pos2">
                <valeur label="g" comment="général"/>
                <valeur label="p" comment="particule"/>
            </trait>
            <trait label="type" comment="pos3">
                <valeur label="c" comment="comparatif"/>
                <valeur label="p" comment="positif"/>
                <valeur label="d" comment="associé à négation"/>
                <valeur label="n" comment="négatif"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="type"/>
        </traits>
    </categorie>

<!-- ****************************************************************************************** -->
    <categorie label="S" comment="préposition">
<!-- ****************************************************************************************** -->

        <traits label="S_traits" comment="traits de la préposition">
            <trait label="soucat" comment="pos2">
                <valeur label="p" comment="préposition"/>
            </trait>
            <trait label="type" comment="pos3">
                <valeur label="a" comment="à"/>
                <valeur label="d" comment="de"/>
            </trait>
            <trait label="composeur" comment="pos4">
                <valeur label="+" comment="composition"/>
                <valeur label="0" comment="pas de composition"/>
            </trait>
            <trait label="categComposeur" comment="pos5">
                <valeur label="D" comment="Déterminant"/>
```

```
                <valeur label="A" comment="Adjectif"/>
                <valeur label="P" comment="Pronom"/>
                <valeur label="0" comment="pas de composition"/>
            </trait>
            <trait label="soucatComposeur" comment="pos6">
                <valeur label="a" comment="Déterminant article?"/>
                <valeur label="d" comment="Déterminant ou Pronom démonstratif"/>
                <valeur label="i" comment="Déterminant ou Adjectif ou Pronom indéfini"/>
                <valeur label="s" comment="Déterminant ou Adjectif ou Pronom possessif"/>
                <valeur label="t" comment="Déterminant ou Pronom interrogatif"/>
                <valeur label="k" comment="Déterminant ou Adjectif ou Pronom cardinal"/>
                <valeur label="f" comment="Adjectif qualificatif"/>
                <valeur label="o" comment="Adjectif ordinal"/>
                <valeur label="p" comment="Pronom personnel"/>
                <valeur label="r" comment="Pronom relatif"/>
                <valeur label="x" comment="Pronom réfléchi"/>
                <valeur label="0" comment="pas de composition"/>
            </trait>
            <trait label="typeComposeur" comment="pos7">
                <valeur label="c" comment="Adjectif comparatif"/>
                <valeur label="p" comment="Adjectif positif"/>
                <valeur label="1" comment="Déterminant ou Pronom première personne"/>
                <valeur label="2" comment="Déterminant ou Pronom deuxième personne"/>
                <valeur label="3" comment="Déterminant ou Pronom troisième personne"/>
                <valeur label="0" comment="pas de composition"/>
            </trait>
            <trait label="genre" comment="pos8">
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
                <valeur label="0" comment="pas de composition"/>
            </trait>
            <trait label="nombre" comment="pos9">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
                <valeur label="0" comment="pas de composition"/>
            </trait>
        </traits>
    </categorie>

<!-- ****************************************************************************************** -->
    <categorie label="V" comment="verbe">
<!-- ****************************************************************************************** -->

        <traits label="V_traits" comment="traits du verbe">
            <trait label="soucat" comment="pos2">
                <valeur label="m" comment="principal"/>
                <valeur label="o" comment="modal"/>
                <valeur label="a" comment="auxiliaire avoir"/>
                <valeur label="e" comment="auxiliaire être"/>
            </trait>
            <trait label="mode" comment="pos3">
                <valeur label="n" comment="infinitif"/>
                <valeur label="i" comment="indicatif"/>
                <valeur label="m" comment="impératif"/>
                <valeur label="c" comment="conditionnel"/>
                <valeur label="s" comment="subjonctif"/>
                <valeur label="p" comment="participe"/>
            </trait>
            <trait label="temps" comment="pos4">
                <valeur label="p" comment="présent"/>
                <valeur label="s" comment="passé"/>
                <valeur label="i" comment="imparfait"/>
                <valeur label="f" comment="futur"/>
            </trait>
            <trait label="ordre" comment="pos5">
                <valeur label="1" comment="première personne"/>
                <valeur label="2" comment="deuxième personne"/>
                <valeur label="3" comment="troisième personne"/>
            </trait>
            <trait label="nombre" comment="pos6">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <trait label="genre" comment="pos7">
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="mode"/>
            <propagation label="temps"/>
            <propagation label="ordre"/>
            <propagation label="nombre"/>
            <propagation label="genre"/>
        </traits>
    </categorie>

<!-- ****************************************************************************************** -->
    <categorie label="NV" comment="Noyau Verbal">
<!-- ****************************************************************************************** -->

        <proprietes label="proprietes" comment="">
            <propriete label="obligation">
                <clause>
                    <membre>
                        <refCateg label="V" comment="verbe"/>
                    </membre>
                </clause>
            </propriete>

            <propriete label="facultativite">
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                    <refValeur label="typePronom" operateur="=" valeur="j" comment="accusatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                    <refValeur label="typePronom" operateur="=" valeur="d" comment="datif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="general"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                <refValeur label="type" operateur="=" valeur="n" comment="negation"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>

            <propriete label="unicite">
                <clause>
                    <membre>
```

```
                        <refCateg label="V" comment="verbe"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                    <refValeur label="typePronom" operateur="=" valeur="j" comment="accusatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                    <refValeur label="typePronom" operateur="=" valeur="d" comment="datif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                                    <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>

            <propriete label="exigence">
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                    <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="V" comment="verbe">
                                    <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                    <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="V" comment="verbe">
                                    <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                </refCateg>
            </membre>
        </clause>
    </propriete>

    <propriete label="exclusion">
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="i" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="i" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="i" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="m" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="m" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
```

```
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="m" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="c" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="c" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="c" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="s" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="s" comment="non infinitif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adverbe">
```

```
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="s" comment="non infinitif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="p" comment="non infinitif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="p" comment="non infinitif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="V" comment="verbe">
                        <refValeur label="mode" operateur="=" valeur="p" comment="non infinitif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
</propriete>

<propriete label="linearite">
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="a" comment="accusatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="d" comment="datif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="a" comment="accusatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="d" comment="datif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
        <membre>
                <refCateg label="V" comment="verbe"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                        <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                <refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                        <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                        <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                        <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                        <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
</propriete>

<propriete label="dependance" comment="modif TV: il manque les OU">
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="n" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="i" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="m" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
```

```
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="c" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="s" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="n" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="i" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
    </clause>
    <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
        <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="m" comment="pas un participe"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                    <refTrait label="ordre"/>
                </refCateg>
        </membre>
```

```
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="c" comment="pas un participe"/>
                <refTrait label="ordre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                <refTrait label="ordre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="s" comment="pas un participe"/>
                <refTrait label="ordre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                <refTrait label="ordre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="n" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="i" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="m" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
```

```
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="c" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="s" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                <refValeur label="typePronom" operateur="=" valeur="n" comment="nominatif"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="n" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="i" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="m" comment="pas un participe"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
    <membre>
            <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                <refTrait label="nombre"/>
            </refCateg>
    </membre>
</clause>
<clause comment="ajout TV: les verbes au participe ne sont pas concernés">
    <membre>
            <refCateg label="V" comment="verbe">
                <refValeur label="mode" operateur="=" valeur="c" comment="pas un participe"/>
```

```
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause comment="ajout TV: les verbes au participe ne sont pas concernés">
                    <membre>
                            <refCateg label="V" comment="verbe">
                                <refValeur label="mode" operateur="=" valeur="s" comment="pas un participe"/>
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>

<!-- ************************************************************************************** -->
    <categorie label="GN" comment="groupe nominal">
<!-- ************************************************************************************** -->
        <traits label="N_traits" comment="traits du nom">
            <trait label="soucat" comment="pos2">
                <valeur label="c" comment="commun"/>
                <valeur label="d" comment="propre avec déterminant"/>
                <valeur label="p" comment="propre sans déterminant"/>
                <valeur label="l" comment="latin?"/>
                <valeur label="k" comment="cardinal"/>
            </trait>
            <trait label="genre" comment="pos3">
                <valeur label="m" comment="masculin"/>
                <valeur label="f" comment="féminin"/>
            </trait>
            <trait label="nombre" comment="pos4">
                <valeur label="s" comment="singulier"/>
                <valeur label="p" comment="pluriel"/>
            </trait>
            <trait label="sigle" comment="pos5">
                <valeur label="s" comment="sigle (abréviation, etc.)"/>
            </trait>
            <trait label="typeNomPropre" comment="pos6">
                <valeur label="c" comment="pays"/>
                <valeur label="h" comment="habitants"/>
                <valeur label="s" comment="société"/>
            </trait>
            <propagation label="soucat"/>
            <propagation label="genre"/>
            <propagation label="nombre"/>
            <propagation label="sigle"/>
            <propagation label="typeNomPropre"/>
        </traits>
        <proprietes label="proprietes">
            <propriete label="obligation">
                <clause>
                    <membre>
                        <operationLogique label="OU">
                            <refCateg label="N" comment="nom"/>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                <refValeur label="typePronom" operateur="=" valeur="o" comment="oblique"/>
                            </refCateg>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="d" comment="démonstratif"/>
                            </refCateg>
                            <refCateg label="P" comment="pronom">
```

```
                                <refValeur label="soucat" operateur="=" valeur="i" comment="indéfini"/>
                            </refCateg>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="r" comment="relatif"/>
                            </refCateg>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="s" comment="possessif"/>
                            </refCateg>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="t" comment="interrogatif"/>
                            </refCateg>
                            <refCateg label="A" comment="adjectif"/>
                        </operationLogique>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">

                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                                <refValeur label="typePronom" operateur="=" valeur="o" comment="oblique"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="d" comment="démonstratif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="i" comment="indéfini"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="r" comment="relatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="s" comment="possessif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="t" comment="interrogatif"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>
            <propriete label="facultativite">
                <clause>
                    <membre>
                            <refCateg label="D" comment="det"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                                <refValeur label="soucat" operateur="=" valeur="p" comment="propre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                <refValeur label="soucat" operateur="#" valeur="p" comment="non particule"/>
                                <refValeur label="type" operateur="#" valeur="n" comment="non négatif"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>

            <propriete label="exigence">
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif"/>
                    </membre>
                    <membre>
                        <operationLogique label="OU">
                            <refCateg label="D" comment="det"/>
                            <refCateg label="N" comment="nom"/>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                            </refCateg>
                        </operationLogique>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adj"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adj"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adj"/>
                    </membre>
                </clause>
            </propriete>

            <propriete label="exclusion">
```

```
    <clause>
        <membre>
                <refCateg label="N" comment="nom">
                    <refValeur label="soucat" operateur="=" valeur="c" comment="non propre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="N" comment="nom">
                    <refValeur label="soucat" operateur="=" valeur="d" comment="non propre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="N" comment="nom">
                    <refValeur label="soucat" operateur="=" valeur="l" comment="non propre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="N" comment="nom">
                    <refValeur label="soucat" operateur="=" valeur="k" comment="non propre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
</propriete>

<propriete label="linearite">
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="A" comment="adjectif"/>
        </membre>
    </clause>
```

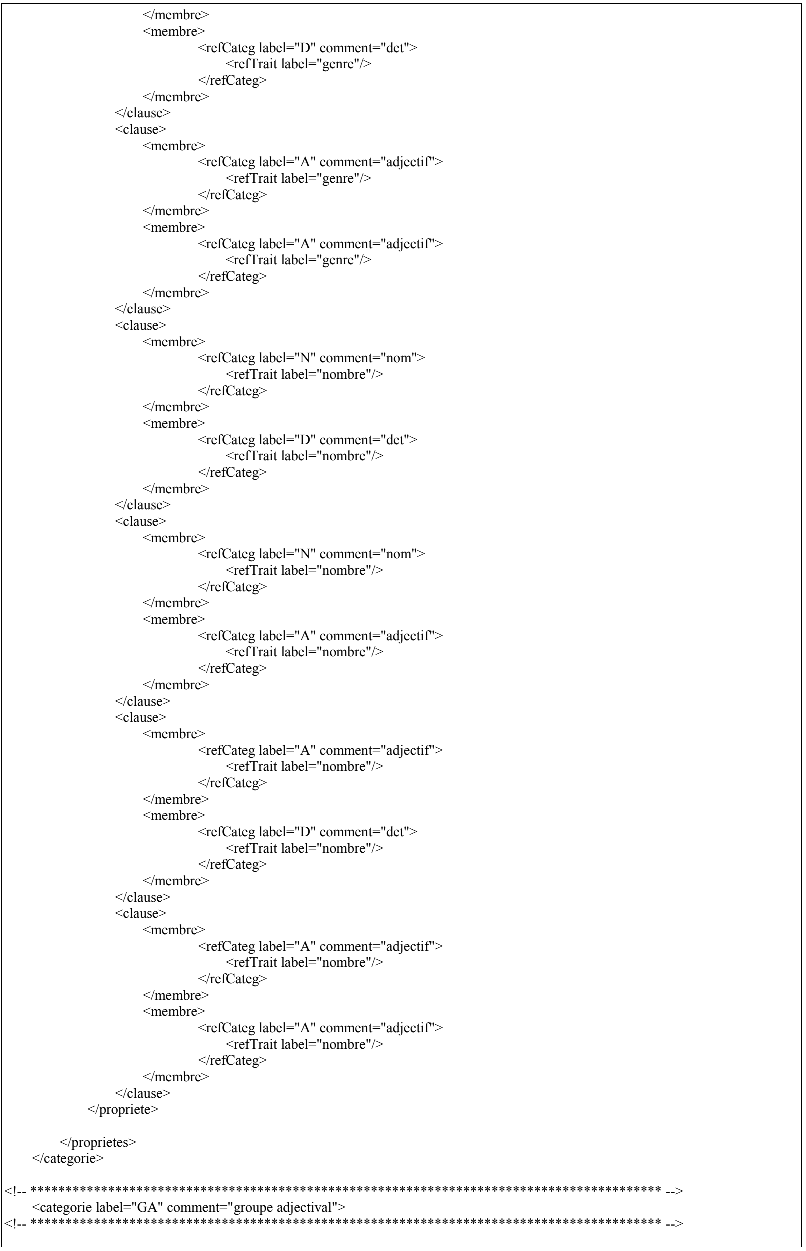
```
                <clause>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="N" comment="nom"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adverbe"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adjectif"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe"/>
                    </membre>
                    <membre>
                            <refCateg label="N" comment="nom"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif"/>
                    </membre>
                    <membre>
                            <refCateg label="N" comment="nom"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adverbe"/>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adj"/>
                    </membre>
                </clause>
            </propriete>

            <propriete label="dependance">
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                                <refTrait label="genre"/>
```

```
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="D" comment="det">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adj">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adj">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="D" comment="det">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adj">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="S" comment="préposition">
                                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adj">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adj">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
          <membre>
              <refCateg label="D" comment="det">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="N" comment="nom">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
          <membre>
              <refCateg label="S" comment="préposition">
                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="N" comment="nom">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
          <membre>
              <refCateg label="A" comment="adj">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="A" comment="adj">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
          <membre>
              <refCateg label="D" comment="det">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="A" comment="adj">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
          <membre>
              <refCateg label="S" comment="préposition">
                <refValeur label="categComposeur" operateur="=" valeur="D" comment="composee avec
déterminant"/>
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="A" comment="adj">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
          <membre>
              <refCateg label="A" comment="adj">
                <refTrait label="nombre"/>
              </refCateg>
          </membre>
        </clause>
      </propriete>
    </proprietes>
```

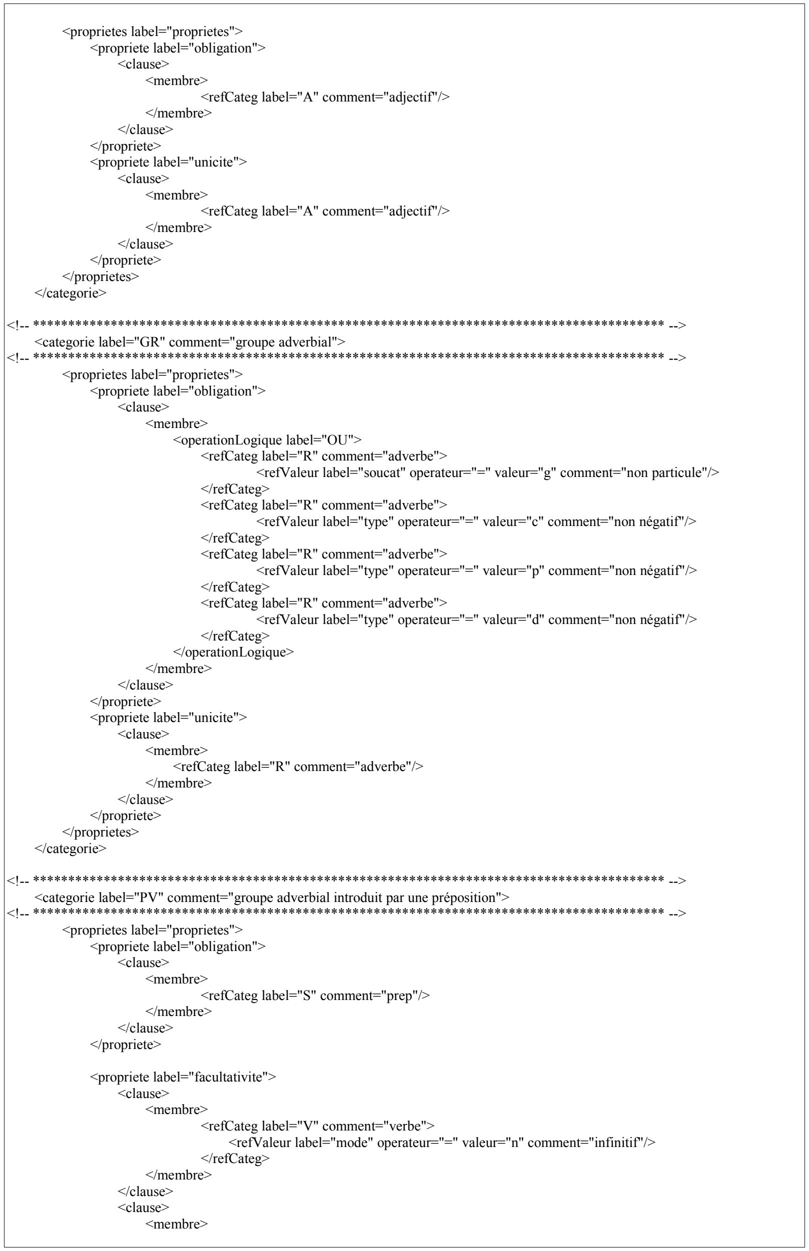
```
  </categorie>

<!-- ************************************************************************************** -->
  <categorie label="GP" comment="groupe prépositionnel">
<!-- ************************************************************************************** -->

    <proprietes label="proprietes">
      <propriete label="obligation">
        <clause>
          <membre>
            <operationLogique label="OU">
              <refCateg label="S" comment="préposition"/>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="r" comment="relatif"/>
                <refValeur label="typePronom" operateur="=" valeur="o" comment="oblique"/>
              </refCateg>
            </operationLogique>
          </membre>
        </clause>
      </propriete>

      <propriete label="facultativite">
        <clause>
          <membre>
              <refCateg label="N" comment="nom">
                <refValeur label="soucat" operateur="#" valeur="p" comment="nom non propre"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                <refValeur label="typePronom" operateur="=" valeur="o" comment="oblique"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="d" comment="démonstratif"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="i" comment="indéfini"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="r" comment="relatif"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="s" comment="possessif"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="P" comment="pronom">
                <refValeur label="soucat" operateur="=" valeur="t" comment="interrogatif"/>
              </refCateg>
          </membre>
        </clause>
        <clause>
          <membre>
              <refCateg label="A" comment="adj"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="det"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="#" valeur="p" comment="non particule"/>
                    <refValeur label="type" operateur="#" valeur="n" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
</propriete>

<propriete label="unicite">
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="o" comment="oblique"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="d" comment="démonstratif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="i" comment="indéfini"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="r" comment="relatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="s" comment="possessif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="t" comment="interrogatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="det"/>
        </membre>
```

```
    </clause>
</propriete>

<propriete label="exigence" comment="correction v.1.5: suppression de N#p=>D">
    <clause>
        <membre>
                <refCateg label="A" comment="adj"/>
        </membre>
        <membre>
            <operationLogique label="OU">
                <refCateg label="D" comment="det"/>
                <refCateg label="N" comment="nom"/>
            </operationLogique>
        </membre>
    </clause>
                    <clause comment="correction v.1.4">
        <membre>
                <refCateg label="S" comment="prep"/>
        </membre>
        <membre>
            <operationLogique label="OU">
                <refCateg label="N" comment="nom"/>
                <refCateg label="P" comment="pronom"/>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </operationLogique>
        </membre>
    </clause>
</propriete>

<propriete label="exclusion">
    <clause>
        <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="r" comment="relatif"/>
                    <refValeur label="typePronom" operateur="=" valeur="o" comment="oblique"/>
                </refCateg>
        </membre>
        <membre>
            <operationLogique label="OU">
                <refCateg label="N" comment="nom"/>
                <refCateg label="P" comment="pronom"/>
                <refCateg label="A" comment="adjectif"/>
                <refCateg label="D" comment="déterminant"/>
                <refCateg label="R" comment="adverbe">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
                <refCateg label="R" comment="adverbe">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
                <refCateg label="R" comment="adverbe">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </operationLogique>
        </membre>
    </clause>
</propriete>

<propriete label="linearite">
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="P" comment="pronom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="A" comment="adjectif"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="S" comment="préposition"/>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
```

```
                <refCateg label="P" comment="pronom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="D" comment="déterminant"/>
        </membre>
        <membre>
                <refCateg label="A" comment="adjectif"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="A" comment="adjectif"/>
        </membre>
        <membre>
                <refCateg label="N" comment="nom"/>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="R" comment="adverbe"/>
        </membre>
        <membre>
                <refCateg label="A" comment="adjectif"/>
        </membre>
    </clause>
</propriete>

<propriete label="dependance">
    <clause>
        <membre>
                <refCateg label="N" comment="nom">
                    <refTrait label="genre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="D" comment="det">
                    <refTrait label="genre"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="N" comment="nom">
                    <refTrait label="genre"/>
                </refCateg>
        </membre>
        <membre>
                <refCateg label="A" comment="adjectif">
                    <refTrait label="genre"/>
                </refCateg>
        </membre>
    </clause>
    <clause>
        <membre>
                <refCateg label="A" comment="adjectif">
                    <refTrait label="genre"/>
                </refCateg>
```



```
                    </membre>
                    <membre>
                            <refCateg label="D" comment="det">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adjectif">
                                <refTrait label="genre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="D" comment="det">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="N" comment="nom">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adjectif">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="D" comment="det">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="A" comment="adjectif">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>

        </proprietes>
    </categorie>

<!-- **************************************************************************************** -->
    <categorie label="GA" comment="groupe adjectival">
<!-- **************************************************************************************** -->
```



```
        <proprietes label="proprietes">
            <propriete label="obligation">
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif"/>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">
                <clause>
                    <membre>
                            <refCateg label="A" comment="adjectif"/>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>

<!-- **************************************************************************************** -->
    <categorie label="GR" comment="groupe adverbial">
<!-- **************************************************************************************** -->
        <proprietes label="proprietes">
            <propriete label="obligation">
                <clause>
                    <membre>
                        <operationLogique label="OU">
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                            </refCateg>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                            </refCateg>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                            </refCateg>
                            <refCateg label="R" comment="adverbe">
                                    <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                            </refCateg>
                        </operationLogique>
                    </membre>
                </clause>
            </propriete>
            <propriete label="unicite">
                <clause>
                    <membre>
                        <refCateg label="R" comment="adverbe"/>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>

<!-- **************************************************************************************** -->
    <categorie label="PV" comment="groupe adverbial introduit par une préposition">
<!-- **************************************************************************************** -->
        <proprietes label="proprietes">
            <propriete label="obligation">
                <clause>
                    <membre>
                            <refCateg label="S" comment="prep"/>
                    </membre>
                </clause>
            </propriete>

            <propriete label="facultativite">
                <clause>
                    <membre>
                            <refCateg label="V" comment="verbe">
                                <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="p" comment="participe"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="#" valeur="o" comment="non oblique"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="R" comment="adv"/>
            </membre>
        </clause>
    </propriete>

    <propriete label="unicite">
        <clause>
            <membre>
                <refCateg label="S" comment="prep"/>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="V" comment="verbe"/>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="j" comment="accusatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="p" comment="personnel"/>
                    <refValeur label="typePronom" operateur="=" valeur="d" comment="datif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="P" comment="pronom">
                    <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="R" comment="adv"/>
            </membre>
        </clause>
    </propriete>

    <propriete label="exigence">
        <clause>
            <membre>
                <refCateg label="S" comment="prep"/>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe"/>
```

```
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="R" comment="adv"/>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe"/>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="P" comment="pronom"/>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe"/>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="n" comment="infinitif"/>
                </refCateg>
            </membre>
        </clause>
    </propriete>

    <propriete label="exclusion">
        <clause>
            <membre>
                <refCateg label="V" comment="verbe">
                    <refValeur label="mode" operateur="=" valeur="p" comment="participe"/>
                </refCateg>
            </membre>
            <membre>
                <refCateg label="R" comment="adv">
                    <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                    <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                </refCateg>
            </membre>
        </clause>
        <clause>
            <membre>
```

```
                            <refCateg label="V" comment="verbe">
                                <refValeur label="mode" operateur="=" valeur="p" comment="participe"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="V" comment="verbe">
                                <refValeur label="mode" operateur="=" valeur="p" comment="participe"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>

            <propriete label="linearite">
                <clause>
                    <membre>
                            <refCateg label="S" comment="prep"/>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adv"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="S" comment="prep"/>
                    </membre>
                    <membre>
                            <refCateg label="V" comment="verbe"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="S" comment="prep"/>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adv"/>
                    </membre>
                    <membre>
                            <refCateg label="V" comment="verbe"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adv"/>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom"/>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                                <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="c" comment="non négatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                                <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="p" comment="non négatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="p" comment="particule"/>
                                <refValeur label="type" operateur="=" valeur="n" comment="négatif"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="R" comment="adv">
                                <refValeur label="soucat" operateur="=" valeur="g" comment="non particule"/>
                                <refValeur label="type" operateur="=" valeur="d" comment="non négatif"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="P" comment="pronom"/>
                    </membre>
                    <membre>
                            <refCateg label="V" comment="verbe"/>
                    </membre>
                </clause>
            </propriete>

            <propriete label="dependance">
                <clause>
                    <membre>
                            <refCateg label="V" comment="verbe">
                                <refTrait label="ordre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                                <refTrait label="ordre"/>
                            </refCateg>
                    </membre>
                </clause>
                <clause>
                    <membre>
                            <refCateg label="V" comment="verbe">
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                    <membre>
                            <refCateg label="P" comment="pronom">
                                <refValeur label="soucat" operateur="=" valeur="x" comment="réfléchi"/>
                                <refTrait label="nombre"/>
                            </refCateg>
                    </membre>
                </clause>
            </propriete>
        </proprietes>
    </categorie>
</grammaire>
```

Appendix C

Instructions to Annotators

Évaluation syntaxique : consignes aux évaluateurs

Jean-Philippe Prost

1 Contexte

L'objet de l'opération est d'évaluer la sortie d'un outil informatique d'analyse syntaxique. Cet outil a pour but de construire un arbre syntaxique pour un énoncé quelconque, potentiellement malformé. Cet arbre syntaxique doit être complet (c'est-à-dire avec une racine unique), et constituer une analyse plausible sur le plan syntaxique.

2 Description de la tâche

Une liste de phrases, indépendantes les unes des autres, chacune annotée d'un arbre syntaxique[1], vous est présentée. La question à laquelle vous devez répondre, pour chaque phrase envisagée, est la suivante :

l'arbre syntaxique proposé constitue-t-il une analyse possible et acceptable de la phrase ?

Il est important d'insister sur le fait que la décision en matière d'*acceptabilité* doit se faire, autant que possible, sur des critères *syntaxiques*. L'objectif n'est pas de juger l'acceptabilité de la phrase en soi (la plupart sont intentionellement malformées), mais bien celle de l'analyse syntaxique qui lui est associée. La difficulté réside dans l'évaluation de l'adéquation entre malformation et arbre syntaxique complet.

2.1 Feuille de résultats

La réponse apportée est nécessairement binaire : l'arbre est correct ou ne l'est pas. Un arbre correct est noté 1, tandis qu'un arbre incorrect est noté 0. Vous pouvez reporter les résultats dans la colonne *Correct* de la feuille de résultats jointe. La colonne *Alternative* est décrite § 3. La colonne *Commentaire* est optionnelle. Elle peut vous permettre de joindre une note libre, pour justifier une réponse difficile, soulever ce qui vous semble être une anomalie, ou simplement faire part d'une remarque.

2.2 Catégories lexicales et syntaxiques

Les catégories lexicales et syntaxiques utilisées sont basées essentiellement sur le système d'annotation MULTEXT, à quelques détails prêts. Les catégories que vous rencontrerez sont les suivantes :

D Déterminant

[1] Certaines phrases sont en fait également associées à une liste d'alternatives ; seule la solution principale, présentée sur fond blanc, fait l'objet de cette évaluation.

N	Nom
V	Verbe
A	Adjectif
R	Adverbe
C	Conjonction
P	Pronom
S	Préposition
GN	Groupe Nominal
NV	Noyau Verbal
GA	Groupe Adjectival
GP	Groupe Prépositionnel
CN	Coordination Nominale[2]
Ut	Phrase (pour *Utterance*, en anglais)
Star	*catégorie "joker", représentant l'absence d'une racine unique*

2.3 Consignes supplémentaires

1. Un arbre dont la pseudo-racine est *Star* est nécessairement *incorrect*.
2. Les catégories lexicales sont considérées comme correctes. Certaines anomalies peuvent cependant subsister. Dans ce cas, l'arbre doit être considéré comme incorrect. Une note signalant le cas dans la colonne *Commentaire* de la feuille de résultats sera la bienvenue.

3 Tâche subsidiaire

Lorsque l'arbre principal est jugé incorrect, et qu'une liste d'alternatives est proposée (arbres sur fond coloré), indiquer dans la colonne *Alternative* si une solution correcte y figure (valeur: 1). La réponse par défaut est négative (i.e. aucune alternative correcte), il n'est donc pas nécessaire de renseigner ce cas. Il est inutile également de parcourir les alternatives lorsque l'arbre principal est correct.

4 Remarques complémentaires

- Le corpus qui vous est présenté comporte une centaine de phrases, ce qui devrait demander environ 1h à évaluer. Cependant, le facteur temps n'ayant aucune importance, vous pouvez décider de vous arrêter lorsque vous le souhaitez—voire même réclamer un corpus supplémentaire !
- Les phrases sont numérotées ; il est normal que certains numéros manquent.

[2] L'existence de la *Coordination Nominale* n'est justifiée que pour des raisons techniques. Elle doit simplement être vue comme un groupe nominal englobant une coordination.