HAL
open science

Modeling Preferences for Ambiguous Utterance Interpretations

Mehdi Mirzapour

To cite this version:

Mehdi Mirzapour. Modeling Preferences for Ambiguous Utterance Interpretations. Other [cs.OH]. Université Montpellier, 2018. English. NNT : 2018MONTS094 . tel-01908642v2

HAL Id: tel-01908642

https://hal.science/tel-01908642v2

Submitted on 8 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Informatique
École doctorale Information, Structures et Systèmes

Unité de recherche LIRMM

Modeling Preferences for Ambiguous Utterance Interpretations

Présentée par Mehdi MIRZAPOUR

Le 28 septembre 2018

Sous la direction de Pr. Christian RETORÉ et Dr. Jean-Philippe PROST

Devant le jury composé de

Pr Veronica DAHL, Professeur, Université de Simon Fraser	Rapporteur
Pr Christophe FOUQUERÉ, Professeur, Université Paris 13, LIPN	Rapporteur
Dr Maxime AMBLARD, MCF HDR, Université de Lorraine LORIA	Examinateur
Dr Philippe BLACHE, Directeur de Recherche, CNRS LPL	Examinateur
Pr Violaine PRINCE, Professeur, Université de Montpellier LIRMM	Examinateur
Dr Jean-Philippe PROST, MCF, Université de Montpellier LIRMM	Co-directeur
Pr Christian RETORÉ, Professeur, Université de Montpellier LIRMM	Directeur

UNIVERSITÉ DE MONTPELLIER

Declaration of Authorship

I, Mehdi MIRZAPOUR, declare that this thesis titled, "Modeling Preferences for Ambiguous Utterance Interpretations" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Like a sword, a word can wound or kill, but as long as one does not touch the blade, the sword is no more than a smooth piece of metal. Someone who knows the qualities of a sword does not play with it, and someone who knows the nature of words does not play with them."

Miyamoto Musashi

UNIVERSITY OF MONTPELLIER

Abstract

Computer Science
LIRMM

Doctor of Philosophy

Modeling Preferences for Ambiguous Utterance Interpretations

by Mehdi MIRZAPOUR

The problem of automatic logical meaning representation for ambiguous natural language utterances has been the subject of interest among the researchers in the domain of computational and logical semantics. Ambiguity in natural language may be caused in lexical/syntactical/semantical level of the meaning construction or it may be caused by other factors such as ungrammaticality and lack of the context in which the sentence is actually uttered. The traditional Montagovian framework and the family of its modern extensions have tried to capture this phenomenon by providing some models that enable the automatic generation of logical formulas as the meaning representation. However, there is a line of research which is not profoundly investigated yet: to rank the interpretations of ambiguous utterances based on the real preferences of the language users. This gap suggests a new direction for study which is partially carried out in this dissertation by modeling meaning preferences in alignment with some of the well-studied human preferential performance theories available in the linguistics and psycholinguistics literature.

In order to fulfill this goal, we suggest to use/extend Categorial Grammars for our syntactical analysis and Categorial Proof Nets as our syntactic parse. We also use Montagovian Generative Lexicon for deriving multi-sorted logical formula as our semantical meaning representation. This would pave the way for our five-fold contributions, namely, (i) ranking the multiple-quantifier scoping by means of underspecified Hilbert's epsilon operator and categorial proof nets; (ii) modeling the semantic gradience in sentences that have implicit coercions in their meanings. We use a framework called Montagovian Generative Lexicon. Our task is introducing a procedure for incorporating types and coercions using crowd-sourced lexical data that is gathered by a serious game called JeuxDeMots; (iii) introducing a new locality-based referent-sensitive metrics for measuring linguistic complexity by means of Categorial Proof Nets; (iv) introducing algorithms for sentence completions with different linguistically motivated metrics to select the best candidates; (v) and finally integration of different computational metrics for ranking preferences in order to make them a unique model.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Christian Retoré and Jean-Philippe Prost for being willing to take me as a student and also for their continuous and extraordinary support of my Ph.D study, for their patience, motivation, and immense knowledge. I will be forever grateful for their help.

Besides my advisors, I would like to appreciate Philippe Blache and Richard Moot for their kind support and the time that they spent to discuss novel ideas. I am profoundly influenced by their research style and wise approaches. My sincere thanks also go to Violaine Prince, Mathieu Lafourcade, Bruno Mery, Davide Catta and Alain Joubert who provided patiently discussions and hints during my research in TEXTE group. I would like to express my gratitude for Jan van Eijck and Christina Unger for introducing their invaluable book entitled *Computational Semantics with Functional Programming* which inspired me to enter to the field of Computational Semantics.

I would like to thank my Ph.D. committee members Maxime Amblard, Philippe Blache, Veronica Dahl, Christophe Fourqueré and Violaine Prince for their insightful comments and constructive ideas which deserve further attention for my future research and also my internal supervision committee Madalina Croitoru, David Delahaye and I2S doctoral school responsible Marianne Huchard. I appreciate very supportive and professional administration members of our lab Elisabeth Greverie, Cecile Lukasik, Guylaine Martinoty and Nicolas Serrurier. I thank my fellow labmates Jimmy Benoits, Davide Catta, Maxime Chapuis, Nadia Clairet, Kévin Cousot, Lionel Ramadier and Noémie-Fleur Sandillon-Rezer for their stimulating discussions. Also, I thank my ex-supervisors Gholamreza Zakiani and Fereshteh Nabati for paving the way for my Ph.D.with providing support and encouragement for the continuation of my study before and after starting my PhD. I am grateful to Fereydoon Fazilat for enlightening the first glance of logical study. I am also grateful to Gyula Klima for sharing his insightful ideas on logic, language and semantics and also for his kind support of my Ph.D. applications.

I would also like to thank the following people for their support, ideas and lectures that have influenced me: Ali-Akbar Ahmadi Aframjani, Fabio Alves, Lasha Abzianidze, Jean-Yves Béziau, Patrick Blackburn, Johan Bos, Adrian Brasoveanu, Michael Carl, John Corcoran, Stergios Chatzikyriakidis, Robin Cooper, Alexander Dikovsky, Jakub Dotlačil, George Englebretsen, Edward Gibson, Jean-Yves Girard, Philippe De Groote, Hans Götzsche, Mohammad-Ali Hojati, Rodrigo Guerizoli, Herman Hendriks, Mark Johnson, Joachim Lambek, Anaïs Lefeuvre, Victoria Lei, Defeng Li, Roussanka Loukanova, Zhaohui Luo, Laura Kallmeyer, Vedat Kamer, Ron Kaplan, Amirouche Moktefi, Friederike Moltmann, Glyn Morrill, Larry Moss, Reinhard Muskens, Lotfollah Nabavi, Katashi Nagao, Petya Osenova, Mohammadreza A. Oskoei, Terence Parsons, Wiebke Petersen, James Pustejovsky, Aarne Ranta, Stephen Read, Mehrnoosh Sadrzadeh, Moritz Schaeffer, Jeremy Seligman, Kiril Simov, Mark Steedman, Jakub Szymanik, Simon Thompson, Erik Thomsen, Şafak Ural, Shravan Vasishth and Yorick Wilks.

Last but not the least, I would like to thank my wife Fatemeh, my children Iliya and Mélodie, my parents, my sisters and my friends for supporting me spiritually throughout writing this thesis.

Contents

Declaration of Authorship **iii**

Abstract **vii**

Acknowledgements **ix**

1 Introduction **1**

2 Background Knowledge **5**
2.1 Lambek Calculus 5
2.2 Categorial Proof Nets 6
2.3 Montagovian Generative Lexicon 13
2.4 JeuxdeMots : A Lexical-Semantic Network 17

3 Modeling Meanings Preferences I: Ranking Quantifier Scoping **21**
3.1 Introduction 21
3.2 Gibson's Incomplete Dependency Theory 22
3.3 Incomplete Dependency Complexity Profiling, and its limitations . . 23
3.3.1 Formal Definitions and Example 23
3.3.2 Objection I: A Minor Problem 25
3.3.3 Objection II: A Major Problem 26
3.4 Hilbert's Epsilon, Reordering Cost and Proof-nets: A New Model . . 33
3.4.1 In situ (=Overbinding) Quantification 33
3.4.2 Quantifiers Order Measurement 34
3.4.3 Examples 35
3.5 Limitations 36
3.6 Conclusion and Possible Extensions 36

4 Modeling Meanings Preferences II: Lexical Meaning and Semantic Gradience **37**
4.1 Introduction 37
4.2 The Lexicon Requirements 38
4.3 Lexical Data Crowd-Sourced from Serious Games 40
4.3.1 Lexemes, Sorts and Sub-Types 41
4.3.2 Lexical Transformations 42
4.3.3 Adapting the Argument 42
Argument-driven transformations 43
Predicate-driven transformations 43
4.3.4 Adapting the Predicate 44
4.3.5 Constraints and Relaxation 44
4.4 Integrating and Ranking Transformations 44
4.4.1 Adding Collected Transformations to the Lexicon 44
4.4.2 Scoring Interpretations 45
4.4.3 Correcting the Lexicon using Different Sources 46

4.5 Preference Mechanism for Quantifying Semantic Gradience 46
4.5.1 Preference-as-procedure v.s. Preference-as-restriction 46
4.5.2 Case Study 47
The Straightforward Case 47
Lexicon Organization in MGL and Meaning Representation . . 48
Collecting Coercions 48
The Non-Human Case 49
Limits of MGL 50
A Direct Solution 50
4.6 Conclusion and Future Works 51

5 Modeling Meanings Preferences III: Categorial Proof Nets and Linguistic Complexity 53
5.1 Introduction 53
5.2 Gibson's Theories on Linguistic Complexity 55
5.2.1 Incomplete Dependency Theory 55
5.2.2 Dependency Locality Theory 56
5.3 Incomplete Dependency-Based Complexity Profiling and its Limitation 56
5.3.1 Formal Definitions and Example 56
5.3.2 Limitation 57
5.4 A New Proposal: Distance Locality-Based Complexity Profiling 60
5.5 Evaluation of the New Proposal against other Linguistic Phenomena 61
5.6 Limitations 68
5.7 Conclusion and Possible Extensions 75

6 Modeling Meanings Preferences IV: Ranking Incomplete Sentence Interpretations 79
6.1 Introduction 79
6.2 Sentence Completion: Algorithm A 81
6.2.1 Definitions 81
6.2.2 Unification Technique of RG Algorithm 83
6.2.3 AB grammars, Unification, and Dynamic Programming 85
6.2.4 Algorithm A 87
6.2.5 Limitations of Algorithm A 87
6.3 Sentence Completion: Algorithm B 88
6.3.1 Syntax and Semantics of Constraint Handling Rules 88
6.3.2 Converting AB Grammar to CFG in Chomsky Normal Form . 89
6.3.3 Algorithm B: Syntax of the Constraint Rules 89
6.3.4 Algorithm B: Semantics of the Constraint Rules 92
6.3.5 Properties of the Algorithm B 95
6.3.6 Limitations of Algorithm B 96
6.4 Ranking Interpretations by Means of Categorial Proof Nets 96
6.4.1 Quantifying Preferences with Distance-based Theories 97
6.4.2 Quantifying Preferences with Activation Theory 100
Definitions of the Activation-based Measuring 101
6.4.3 Quantifying Preferences with Satisfaction Ratio 104
Definitions of the Satisfaction Ratio 107
6.5 Conclusion and Future Works 107

7 Putting It All Together: Preference over Linguistics Preferences **115**
7.1 Introduction . . . 115
7.2 Complexity Metrics: Summaries . . . 115
7.2.1 Quantifiers Order Measurement . . . 115
7.2.2 Incomplete Dependency Theory . . . 116
7.2.3 Dependency Locality Theory . . . 116
7.2.4 Activation Theory . . . 117
7.2.5 Satisfaction Ratio . . . 118
7.3 Integration of Linguistic Difficulty Metrics . . . 118
7.3.1 Integrating Dependency Locality and Incomplete Dependency Theories . . . 118
7.3.2 Integrating Incomplete Dependency and Activation Theories . 119
7.3.3 Integration of the Satisfaction Ratio . . . 123
7.4 Preference over Linguistic Preferences: A General Scheme . . . 123
7.5 Some Examples . . . 124
7.6 Limitations . . . 126
7.7 Conclusion . . . 127

8 Conclusion **129**
8.1 Summaries . . . 130
8.2 Future Works . . . 130
8.2.1 Ontological-based Modeling of the Quantifier Scoping Preference . . . 130
8.2.2 Annotated Data-set for Language Meaning Preferences . . . 132
8.2.3 Integrating Multiple Coercions in Montagovian Generative Lexicon . . . 132
8.2.4 New Metrics for Linguistic Meaning Complexity . . . 132
8.2.5 Enhancing Sentence Completion Algorithms . . . 132

A Mathematical Proofs of the Properties of Algorithm B **135**
A.1 Proof of the Property (A) . . . 135
A.2 Proof of the Property (B) . . . 136
A.3 Proof of the Property (C) . . . 137
A.4 Proof of the Property (D) . . . 137
A.5 Proof of the Property (E) . . . 138
A.6 Proof of the Property (F) . . . 138
A.7 Proof of the Property (G) . . . 139
A.8 Proof of the Property (H) . . . 139
A.9 Proof of the Property (I) . . . 140
A.10 Proof of the Property (J) . . . 142

B Detailed Calculations of the Linguistic Complexity Metrics **145**
B.1 Examples Related to the Chapter 3 . . . 145
B.2 Examples Related to the Chapter 5 . . . 145
B.3 Examples Related to the Chapter 6 . . . 147
B.4 Examples Related to the Chapter 7 . . . 147

C Published Work **149**
C.1 PhD Publications . . . 149
C.2 Other Publications . . . 150

Bibliography **153**

List of Figures

2.1 Local Trees of the Polar Category Formulas 7
2.2 The polar categorial tree of $((np\backslash S)/(np\backslash S))^{\perp}$ 8
2.3 The categorial proof-net analysis for the example "Every barber shaves himself." 9
2.4 The categorial proof-net analysis with labels for the example "Every barber shaves himself." 11
2.5 Polymorphic conjunction: $P(f(x))\&Q(g(x))$ with $x : \xi$, $f : \xi \to \alpha$, $g : \xi \to \beta$. 15
2.6 A part of the JDM network (taken form [LJ15]). 18
2.7 Screen-shot of an ongoing game with the target verb *fromage* (cheese) (taken form [Cha+17a]). 19
2.8 Screen-shot of the game result with the target noun *fromage* (taken from [Cha+17a]). 20

3.1 Proof net analyses for 3.3a (left hand) and 3.3b (right hand) with the relevant profiles. 24
3.2 Proof net analyses for reading (a) of the example 3.4a. 27
3.3 Proof net analyses for reading (b) of the example 3.4b. 28
3.4 Proof net analyses for reading (c) of the example 3.4c. 29
3.5 Proof net analyses for reading (d) of the example 3.4d. 30
3.6 Proof net analyses for reading (e) of the example 3.4e. 31
3.7 Proof net analyses for 3.6a (left hand) and 3.6b (right hand) with the relevant profiles. 32
3.8 New procedure for the example 3.6 35

4.1 Obtaining Coercions: General Scheme, Query Code and Outcome Table for Example (a) 49
4.2 Obtaining Coercions: General Scheme, Query Code and Outcome Table for Example (b) 51

5.1 IDT-based Complexity Profiles for 5.3a and 5.3b. 57
5.2 Proof net analyses for 5.3a (subject-extracted relative clause). 58
5.3 Proof net analyses for 5.3b (object-extracted relative clause). 59
5.4 Accumulative DLT-based Complexity Profiles for 5.3a and 5.3b. 61
5.5 Proof net analysis for the example 5.4a. 62
5.6 Accumulative DLT-based Complexity Profiles for 5.4a and 5.4b. 62
5.7 Proof net analysis for the example 5.4b. 63
5.8 Accumulative DLT-based Complexity Profiles for 5.5a and 5.5b 64
5.10 Proof net analyses for 5.5a located in top (first attempt reading) and 5.5b in bottom (full garden path sentence). 65
5.11 Proof net analyses for 5.6a (object relativization). 66
5.12 Proof net analyses for 5.6b (subject relativization). 67
5.9 Accumulative DLT-based Complexity Profiles for 5.6a and 5.6b 68
5.13 Proof net analyses for 5.7 with highest adverbal attachments. 69

5.14 Proof net analyses for 5.7 with middle adverbal attachments. 70
5.15 Proof net analyses for 5.7 with lowest adverbal attachments. 71
5.16 Accumulative DLT-based complexity profiles for three readings of 5.7 71
5.17 Proof net analyses for 5.8 with sensical (left) and nonsensical (right) interpretations. 72
5.18 Accumulative DLT-based complexity profiles for two readings of 5.8 . 72
5.19 Proof net analyses for 5.9a 73
5.20 Proof net analyses for 5.9b 74
5.21 Accumulative DLT-based complexity profiles for 5.9a and 5.9b 74

6.1 Possible structural trees of a sequence $w_1 w_2 w_3$ 82
6.2 Three possible structural trees for the example 6.2 84
6.3 Proof net analyses for 6.6a and 6.6b. 99
6.4 Proof net analysis for 6.7a. 102
6.5 Proof net analysis for 6.7b. 103
6.6 Proof net analysis for 6.7b with the weights. 105
6.7 Proof net analysis for 6.9a with the (violation) weights. 108
6.8 Proof net analysis for 6.9b with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0' 109
6.9 Proof net analysis for 6.9c with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0' 110
6.10 Proof net analysis for 6.9d with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0' 111
6.11 Proof net analysis for 6.9e with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0'. 112

7.1 Proof net analysis for 7.1a. 120
7.2 Proof net analysis for 7.1b. 121
7.3 The General Scheme for Integrating the Rankings. 122

8.1 Summary of the problems and the relevant solutions. 131

List of Tables

2.1 Sequent calculus rules for LC 6

6.1 Unification of variable categories and fixed (learned) categories 82
6.2 Unification of variable types and learned types 84
6.3 Step (1) of parsing with unification for the example 6.3 85
6.4 Step (2) of parsing with unification for the example 6.3 85
6.5 Step (3) of parsing with unification for the example 6.3 86
6.6 Last step of parsing with unification for the example 6.3 86
6.7 Parsing with unification for the example 6.6 97
6.8 Parsing with unification for the example 6.6 98
6.9 Calculation of the incomplete dependency number for 6.6a and 6.6b. . 98
6.10 Calculation of the dependency locality number for 6.6a and 6.6b. . . . 98
6.11 Calculation of the incomplete dependency number for 6.7a and 6.7b. . 101
6.12 Calculation of the dependency locality number for 6.7a and 6.7b. . . . 101
6.13 Calculation of the activation-based complexity measurement for 6.7a and 6.7b. 104

7.1 Calculation of the incomplete dependency number for 7.1a and 7.1b. . 119
7.2 Calculation of the dependency locality number for 7.1a and 7.1b. . . . 119

B.1 Calculation of the incomplete dependency number for 3.3a and 3.3b. . 145
B.2 Calculation of the incomplete dependency number for 3.6a and 3.6b. . 145
B.3 Calculation of the incomplete dependency number for 5.3a and 5.3b. . 145
B.4 Calculation of the dependency locality number for 5.3a and 5.3b. . . . 145
B.5 Calculation of the dependency locality number for 5.4a and 5.4b. . . . 146
B.6 Calculation of the dependency locality number for 5.5a and 5.5b. . . . 146
B.7 Calculation of the dependency locality number for 5.6a and 5.6b. . . . 146
B.8 Calculation of the dependency locality number for the readings of 5.7. 146
B.9 Calculation of the dependency locality number for the readings of 5.8. 146
B.10 Calculation of the dependency locality number for 5.9a and 5.9b. . . . 146
B.11 Calculation of the incomplete dependency number for 6.6a and 6.6b. . 147
B.12 Calculation of the dependency locality number for 6.6a and 6.6b. . . . 147
B.13 Calculation of the incomplete dependency number for 6.7a and 6.7b. . 147
B.14 Calculation of the dependency locality number for 6.7a and 6.7b. . . . 147
B.15 Calculation of the activation-based complexity measurement for 6.7a and 6.7b. 147
B.16 Calculation of the incomplete dependency number for 7.1a and 7.1b. . 147
B.17 Calculation of the dependency locality number for 7.1a and 7.1b. . . . 148

List of Abbreviations

CG	**C**ategorial **G**rammar
CHR	**C**onstraint **H**andling **R**ules
CPN	**C**ategorial **P**roof **N**et
DLT	**D**istance **L**ocality **T**heory
GSC	**G**entzen **S**equent **C**alculus
GWAP	**G**ames **W**ith **A** **P**urpose
IDT	**I**ncomplete **D**ependency **T**heory
JDM	**J**eux **D**e **M**ots
LC	**L**ambek **C**alculus
MGL	**M**ontagovian **G**enerative **L**exicon
MLL	**M**ultiplicative **L**inear **L**ogic
PG	**P**roperty **G**rammar

To Miyamoto Musashi (1584-1645)

Chapter 1

Introduction

> To know ten thousand things, know one well.
>
> Miyamoto Musashi

Ambiguity is an important linguistic phenomenon which demands a computational treatment in almost all of the subfields of natural language processing, such as Information Retrieval, Sentiment Analysis, Machine Translation, Question-Answering and Natural Language Understanding. The ambiguity may be caused in lexical/ syntactical/ semantical level of meaning construction or it may be caused by other factors such as ungrammaticality or the absence of the context in which the sentence is actually uttered.

In computational formal semantics, the problem of automatic meaning representation for ambiguous natural language utterances has been extensively studied. More specifically, in Montagovian-style formal semantic frameworks one would be able to derive meanings from a given ambiguous text in a systematic and straightforward approach.

A natural, and yet plausible question that can be raised in the computational formal semantic field is that how different possible meaning representations of a written or spoken utterance might be quantitatively ranked in alignment with the well-studied human preferential performance theories.

This question is not just interesting from a computational linguistic or computational psycholinguistic perspective. It is important due to this fact that some computational frameworks over-generate meaning structures and this necessarily suggests a ranking mechanism to rule out some of them. Even if a computational framework properly generates meanings for ambiguous utterances, ranking them as close as possible to the human performance phenomena would be an interesting task. This would provide a quantitative account in favor of introducing new metrics for the future studies in humanizing the natural language understating, i.e. making machines that understand natural language similar to humans. Those machines are supposed to obtain the meaning of linguistically ambiguous utterances in a way that makes the communication between human and machines smoothly. Obviously, the problem of the misunderstanding between machines and humans would increase when no human-like computational ranking is available.

The main question of this dissertation is how to fill the mentioned gap by providing meaning preferential models for ambiguous utterances in alignment with some of the well-studied human preferential performance theories available in the linguistics and psycholinguistics literature. There are more specific questions which

are clustered around the main general question, such as: How can the problem of the ranking multiple-quantifier sentence be efficiently addressed? How can we have the preferential models for those human preferences which are constrained by the lexical data? How can we use human knowledge in lexical-semantic networks to address the previous question? What is a suitable and proper representation of the sentence meaning? How would these meaning representations pave the way to adopt the modern computational psycholinguistic theories into our ranking algorithms? How can we rank ambiguous incomplete sentences? How can potential interpretations for incomplete sentences quantitatively be ranked as close as possible to human preferences?

Although, our research objectives cover the linguistic phenomena which, in principle, is tied up with the ambiguity problem, we will go beyond the classical problem of ranking the ambiguous utterances, and we discover novel solutions for other linguistic phenomena such as quantifier scoping, the gradient of semanticality, embedded pronouns, garden pathing, unacceptability of center embedded, preference for lower attachment and passive paraphrases acceptability. In order to fulfill this, we use Categorial Grammars for our syntactical analysis and Categorial Proof Nets as our syntactic parse. We also use Montagovian Generative Lexicon for deriving multi-sorted logical formula as our semantical meaning representation. Thanks to the Curry–Howard isomorphism and the semantic readings on proof nets, the syntax-semantics interface will be quite straightforward. It is worth mentioning that the meaning can be represented as multi-sorted higher-order logical formulas. Also, it is straightforwardly possible to reify the higher-order logical meaning representations in alignment with the event semantics in Davidsonian [Dav67] and neo-Davidsonian proposals [Res67; Par90].

Our hypothesis is that by applying the above methodology we can quantitatively model a number of linguistic performance phenomena. Our computational model supports the human performances that are experimentally verified by modern psycholinguistic theories.

This work is organized as follows:

In chapter 2, we provide an overview of the background knowledge that one needs to understand this dissertation. It starts with Lambek Calculus as our syntactic analysis with the essential definitions and exploring Lambek Sequent Calculus. Then, we highlight the problem of spurious ambiguity that exist in the sequent calculus. After that, we explain how Categorial Proof Nets can overcome the spurious ambiguity problem. We also go through all the needed definitions, technicalities and examples that is needed to grasp categorial proof-net and the semantic readings of it. Then, we explicate Montagovian Generative Lexicon and its lexicon organization. We discuss how representing meaning as multi-sorts higher order logic can help us to capture coercion and co-prediction in the language. We end up this chapter with introducing JeuxdeMots, a French lexical-semantic network. The relevant graph structures and the validation process would be generally overviewed.

In chapter 3, we focus on the problem of ranking the valid logical meanings of a given multiple-quantifier sentence only by considering the syntactic quantifier order. We first report some deficiencies that exist in an existing state-of-the-art technique (for solving the problem of quantifier scope ranking) known as Incomplete Dependency-based Complexity Profiling [Mor00]. One of the main problem with

this approach is that it does not properly support the ranking problem in some of the sentences such as sentence-modifier adverbials, nested sentences and direct speech. We try to fix this problem by defining a new metric which is inspired by Hilbert's epsilon and introducing the notion of the reordering cost. We will see how this gives a correct account in favour of the problematic cases.

In chapter 4, we focus on the semantic gradiences that potentially happens in sentences that have implicit coercions in their linguistic meaning. The aim of this study is to find an automatic treatment for this kind of semantic preferences and to capture it in our modeling. In order to perform such a task we need to have a rich lexical information. Frameworks based on Generative Lexicon theories [Pus91] such as Motagovian Generative Lexicon [Ret14], can have a rich logical representation using a Montague-like compositional process. A crucial problem for these systems is then to have sufficient lexical resources (as a rich lexicon incorporating types and coercions) to function. Our main task in this chapter is introducing a procedure which can build such a lexicon for the Montagovian Generative Lexicon. We will do this task by using crowd-sourced lexical data that is gathered by a serious game which is called *JeuxDeMots*. The frequencies of the lexical occurrences— which is automatically gathered by the game players— would play a key role in our ranking mechanism. Our strategy, following [FW83], would be based on a mechanism called preference-as-procedure. We practice such a strategy for automatic treatment of semantic gradience in our computational modeling.

In chapter 5, we provide a quantitative computational account of why such a sentence has a harder parse than some other sentence, or that one analysis of a sentence is simpler than another one. We take for granted the Gibson's results on human processing complexity. Gibson first studied the notion of the nesting linguistic difficulty [Gib91] through the maximal number of incomplete syntactic dependencies that the processor has to keep track of during the course of processing a sentence. We refer to this theory as Incomplete Dependence Theory (IDT) as coined by Gibson. IDT had some limitations for the referent-sensitive linguistic phenomena, which justified the later introduction of the Syntactic Prediction Locality Theory [Gib98]. A variant of this theory, namely Dependency Locality Theory (DLT), was introduced later [Gib00] to overcome the limitations of IDT against the new linguistic performance phenomena. In the original works, both IDT and DLT use properties of linguistic representations provided in Government-Binding Theory [Cho82]. We provide a new metric which uses (Lambek) Categorial Proof Nets. In particular, we correctly model Gibson's account in his Dependency Locality Theory. The proposed metric correctly predicts some performance phenomena such as structures with embedded pronouns, garden pathing, unacceptability of center embedded, preference for lower attachment and passive paraphrases acceptability. Our proposal extends existing distance-based proposals on Categorial Proof Nets for complexity measurement while it opens the door to include semantic complexity, because of the syntax-semantics interface in categorial grammars. The purpose of developing our computational psycholinguistic model is not solely limited to measuring linguistic complexity.

In chapter 6, we work on incomplete utterances with missing categories. In the first place, we introduce two algorithms for resolving incomplete utterances. The first algorithm is based on AB grammars, Chart-based Dynamic Programming and learning Rigid AB grammars with a limited scope of only one missing category which have $O(n^4)$ time complexity (n is the number of the words in a sentece). The

second algorithm employs Constraint Handling Rules which deals with $k > 1$ missing categories which enable us to find more than one missing categories at cost of the exponential complexity. In the second place, we introduce measurements on the fixed categorial proof nets motivated by Gibson's distance-based theories, Violation Ratio factor and Activation Theory.

In chapter 7, we focus on the problem of integration different computational metrics for ranking preferences in order to make them a unique model. We see this problem similar, to some extent, to the problem of aggregation of the preferences. In other words, we provide a procedure to make preference over different linguistic preferences introduced throughout this dissertation. We mainly use the lexicographical ordering for aggregating the different preferences. We see this computational procedure with a number of running examples. In chapter 8 we draw conclusions and discuss avenues for further works.

A quick survey on the Ph.D. published papers is available in the appendix C.

Chapter 2

Background Knowledge

> You should not have any special fondness for a particular [tool], or anything else, for that matter. Too much is the same as not enough. Without imitating anyone else, you should have as much [tools] as suits you.
>
> Miyamoto Musashi

In this chapter, we introduce the background knowledge that is assumed one would need to understand this dissertation.[1] The underlying concepts and the notations within each exploited framework are described. This should not prevent the non-familiar readers to consult the relevant references in each section for detailed information since this chapter is not meant to explicate briefly each framework in a lengthy manner.[2]

2.1 Lambek Calculus

Lambek calculus(LC) is introduced for the first time by Lambek in his seminal paper [Lam58]. The motivation behind this system is to provide a resource-boundedness logic in favor of syntactical parsing under the slogan 'parsing as a deduction'. We provide an overview of the existing material [MR12b] on Lambek Calculus.

Definition 2.1. The category formulas (L_p) are freely generated from a set of usual syntactical primitive types $P = \{S, np, n, pp, \cdots\}$ by directional divisions, namely the binary infix connectives \ (over), / (under) and • (product) as follows:

$$Lp ::= P \mid (L_p \backslash L_p) \mid (L_p / L_p) \mid (L_p \bullet L_p)$$

Note that $A\backslash B$ and B/A have also some intuitive interpretations. An expression of type $A\backslash B$ (resp. B/A) when it looks for an expression of type A on its left (resp. right) forms a compound expression of type B. An expression of type A followed by an expression B is of type $A \bullet B$, and the product is related to \ and / by the following relations:

$$A\backslash(B\backslash X) = (B \bullet A)\backslash X \qquad (X/A)/B = X/(B \bullet A)$$

[1] We assume that the readers have some familiarity with the Typed Lambda Calculus and Intuitionistic Propositional Calculus. Interested readers can consult [HS80; GTL89] for more details on the subject.

[2] The main material for writing the subsections 2.1-2.3 are [MR12a; Ret14].

Definition 2.2. A sequent, $\Gamma \vdash A$, comprises formula A as succedent and Γ a sequence of finite formulas as antecedent. The table 2.1 shows the rules of LC which is provided in Gentzen Sequence Calculus in which A, B and C are formulas, while Γ, Γ' and Δ are sequences which each contains finite formulas:

$$\frac{\Gamma, B, \Gamma' \vdash C \quad \Delta \vdash A}{\Gamma, \Delta, A\backslash B, \Gamma' \vdash C}\ \backslash h \qquad \frac{A, \Gamma \vdash C}{\Gamma \vdash A\backslash C}\ \backslash i \quad \Gamma \neq \epsilon$$

$$\frac{\Gamma, B, \Gamma' \vdash C \quad \Delta \vdash A}{\Gamma, B/A, \Delta, \Gamma' \vdash C}\ /h \qquad \frac{\Gamma, A \vdash C}{\Gamma \vdash C/A}\ /i \quad \Gamma \neq \epsilon$$

$$\frac{\Gamma, A, B, \Gamma' \vdash C}{\Gamma, A \bullet B, \Gamma' \vdash C}\ \bullet h \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta, \Gamma \vdash A \bullet B}\ \bullet i$$

$$\frac{\Gamma \vdash A \quad \Delta_1, A, \Delta_2 \vdash B}{\Delta_1, \Gamma, \Delta_2 \vdash B}\ cut \qquad \frac{}{A \vdash A}\ axiom$$

TABLE 2.1: Sequent calculus rules for LC

We can define a lexicon, that is a function Lex which maps words to finite sets of types. A sequence of words $w_1, \cdots, w_n$ is of type u whenever there exists for each w_i a type t_i in $Lex(w_i)$ such that there exists a proof for sequent $t_1, \cdots, t_n \vdash u$ by applications of the introduced rules for LC.

Example 2.1. Having the lexicon John:np, loves:$(np\backslash S)/np$ and Mary:np, we can prove that the expression *John loves Mary* is a sentence of type S. To prove this, we need to prove that the sequence of the assigned formulas are of type S, namely the sequent $np, (np\backslash S)/np, np \vdash S$ is provable. Here is the straightforward proof:

$$\frac{\dfrac{\dfrac{}{S \vdash S}\ axiom \quad \dfrac{}{np \vdash np}\ axiom}{np, np\backslash S \vdash S}\ \backslash h \qquad \dfrac{}{np \vdash np}\ axiom}{np, (np\backslash S)/np, np \vdash S}\ /h$$

Example 2.2. The following two proofs are related to the sequent $A\backslash B, B\backslash C \vdash A\backslash C$. As one can observe, there are two equivalent derivations which differ only in the applications of the irrelevant rule ordering. This phenomenon–called spurious ambiguity–needs a treatment which will be described in the next section.

$$\frac{\dfrac{\dfrac{C \vdash C \quad B \vdash B}{B, B\backslash C \vdash C}\ \backslash h \qquad A \vdash A}{A, A\backslash B, B\backslash C \vdash C}\ \backslash h}{A\backslash B, B\backslash C \vdash A\backslash C}\ \backslash i \qquad \frac{\dfrac{\dfrac{B \vdash B \quad A \vdash A}{A, A\backslash B \vdash B}\ \backslash h \qquad C \vdash C}{A, A\backslash B, B\backslash C \vdash C}\ \backslash h}{A\backslash B, B\backslash C \vdash A\backslash C}\ \backslash i$$

2.2 Categorial Proof Nets

In the previous section, we showed that the Gentzen Sequence Calculus has the problem of the spurious ambiguity. One way to overcome this problem is to use the proof net as a vehicle for our logical derivations. Proof nets are the structures that were originally introduced by Girard for linear logic [Gir87]. We use some of the existing adoptions of Girard's system made in favor of Lambek categorial grammar

[MR12b, Chap 6]. The outcome is quite satisfying since we can end up with a logic-based formalism that can linguistically be interpreted as parse structures.

In order to relate the categorial formula (L_p) to linear logic formula, we need to introduce negation denoted by $(\cdots)^{\perp}$ *(the orthogonal of $\cdots$)* and two linear logic connectives, namely multiplicative conjunction denoted by $\otimes$ and multiplicative disjunction denoted by $\parr$. Now we can translate L_p to the linear logic formulas by the following definitions and equivalences:

$$\text{Definition of } \backslash \text{ and } /: \quad A\backslash B \equiv A^{\perp} \wp B \qquad B/A \equiv B \parr A^{\perp}$$
$$\text{De Morgan equivalences } (A^{\perp})^{\perp} \equiv A \qquad (A \parr B)^{\perp} \equiv B^{\perp} \otimes A^{\perp} \qquad (A \otimes B)^{\perp} \equiv B^{\perp} \parr A^{\perp}$$

Definition 2.3. A polar category formula is a Lambek categorial type labeled with positive ($^{\circ}$) or negative ($^{\bullet}$) polarity recursively definable as follows:

$$L^{\circ} ::= P \mid (L^{\bullet} \parr L^{\circ}) \mid (L^{\circ} \parr L^{\bullet}) \mid (L^{\circ} \otimes L^{\circ})$$
$$L^{\bullet} ::= P^{\perp} \mid (L^{\circ} \otimes L^{\bullet}) \mid (L^{\circ} \otimes L^{\bullet}) \mid (L^{\bullet} \parr L^{\bullet})$$

The above formulation allows $\otimes$ of positive formulas and $\parr$ of negative formulas, this would allow us to have $\otimes$ in addition to the $\backslash$ and $/$ symbols in categories. Combining heterogeneous polarities guarantees that a positive formula is a category and that a negative formula is the negation of a category.

Definition 2.4. Based on the previous inductive definitions, we can have an easy decision procedure to check whether a formula F is in L° or $L^{\bullet}$:

$\otimes$	$\bullet$	$\circ$
$\bullet$	*undefined*	$\bullet$
$\circ$	$\bullet$	$\circ$

$\parr$	$\bullet$	$\circ$
$\bullet$	$\bullet$	$\circ$
$\circ$	$\circ$	*undefined*

Example 2.3. $a \parr a$ has no polarity; $a^{\perp} \parr b$ is positive and it is $a\backslash b$ while $b^{\perp} \otimes a$ is negative and it is the negation of $a\backslash b$.

Definition 2.5. A polar category formula tree is a binary ordered tree in which the leaves are labeled with polar atoms and each local tree is one of the following logical links:

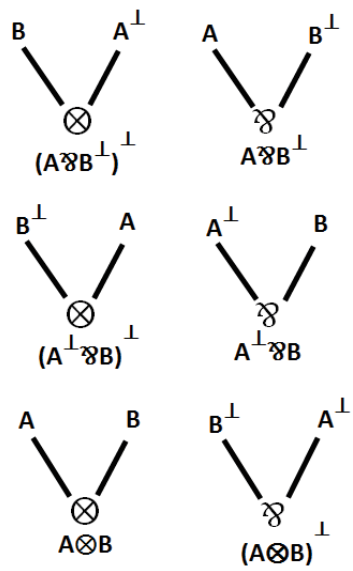


FIGURE 2.1: Local Trees of the Polar Category Formulas

Example 2.4. The figure 2.2 illustrates the polar category formula tree of the formula $((np\backslash S)/(np\backslash S))^{\perp}$ which is the negation of the $(np\backslash S)/(np\backslash S)$. This specific category is a kind of categorial formula that represents auxiliary verbs in the linguistics.

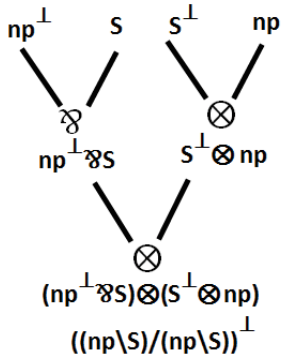


FIGURE 2.2: The polar categorial tree of $((np\backslash S)/(np\backslash S))^{\perp}$

Definition 2.6. A proof frame is a finite sequence of polar category formula trees with one positive polarity corresponding to the unique succedent of sequent.

Definition 2.7. A proof structure is a proof frame with axiom linking which corresponds to the axiom rule in the sequence calculus. Axioms are a set of pairwise disjoint edges connecting a leaf z to a leaf $z^{\perp}$, in such a way that every leaf is incident to some axiom link.

Definition 2.8. A proof net is a proof structure satisfying the following conditions: [3]

Acyclicity: every cycle contains the two edges of the same $\wp$ branching.

Enumerate: there is a path not involving the two edges of the same $\wp$ branching between any two vertices.

Intuitionism: every conclusion can be assigned some polarity.

Non commutativity: the axioms do not cross (are well bracketed).

Example 2.5. The figure 2.3 shows the categorial analysis of the sentence *Every barber shaves himself*. For each word a proper category from the lexicon is assigned. We re-analyze this syntactic analysis in the next example in order to get the semantic meaning represented as a logical formula.

It has been known for many years that categorial parse structures, i.e. proof in some substructural logic, are better described as proof nets [Roo91; Ret96; Moo02; MR12a]. Indeed, categorial grammars following the parsing-as-deduction paradigm, an analysis of a c phrase $w_1, , \ldots, w_n$ is a proof of c under the hypotheses $c_1, ..., c_n$ where c_i is a possible category for the word w_i; and proofs in those systems are better denoted by graphs called proof nets. The reason is that different proofs in the Lambek calculus may represent the same syntactic structure (constituents and dependencies), but these essentially similar sequent calculus proofs correspond to a unique proof net.

[3]This list is redundant: for instance intuitionism plus acyclicity implies connectedness.

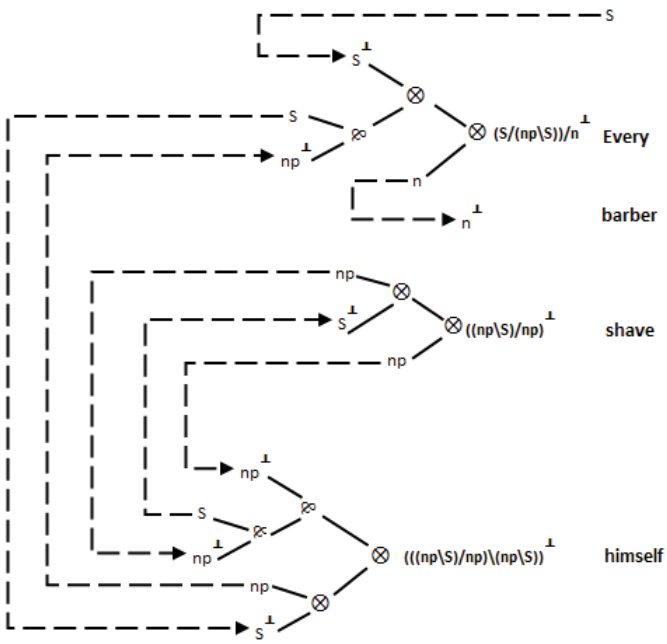


FIGURE 2.3: The categorial proof-net analysis for the example "Every barber shaves himself."

Proof-nets have an important advantage over other representations of categorial analyses: they avoid the phenomenon known as spurious ambiguities, that is when different parse structures correspond to the same syntactic structure (same constituent and dependencies). Indeed proofs (parse structures) with unessential differences are mapped to the same proof net. A (normal) deduction of $c_1, ..., c_n \vdash c$ (i.e. a syntactic analysis of a sequence of words as a constituent of category c) maps to a (normal) proof net with conclusions $(c_n)^\perp, ..., (c_1)^\perp, c$ [Roo91; Roo92; MR12a]. Conversely, every normal proof net corresponds to at least one normal sequent calculus proof [Ret96; MR12a].

Definition 2.9. The semantics associated with a categorial proof net (the proof as a lambda term under the Curry-Howard correspondence) can be gained by associating a distinct index with each par-node and traveling as follows [DGR96]:

- Enter the proof net by going up at the unique output root.
- Follow the path specified by the output polarities until an axiom-link is eventually reached; this path, which is ascending, is made of par-links that correspond to successive lambda-abstractions.
- Cross the axiom-link following the output-input direction.
- Follow the path specified by the input polarities; this path, which is descending, is made of tensor-links that correspond to successive applications; it ends either on some input conclusion of the proof-net, or on the input premise of some par-link; in both cases, the end of the path coincides with the head-variable of the corresponding lambda-term; in the first case (input conclusion) this head-variable is free; in the second case (premise of a par-link) this head-variable is bound to the lambda corresponding to the par-link.
- In order to get all the arguments to which the head-variable is applied, start again the same sort of traversal from every output premise of the tensor-links that have been encountered during the descending phase described in the previous section.

Example 2.6. Figure 2.4 shows the same proof-net illustrated in 2.3 for the sentence *Every barbers shave himself*. The only difference is the labels that are sketched in the figure in order to show each single step for deriving the semantic reading of the proof net as described in the definition 2.9. Here are the details:

$$T_1 = T_2 T_3$$

$$T_2 = every(barber)$$

$$T_3 = \lambda x_1.T_4$$

$$T_4 = T_5 x_1$$

$$T_5 = himself(T_6)$$

$$T_6 = \lambda x_2.T_7$$

$$T_7 = \lambda x_3.T_8$$

$$T_8 = T_9(x_3)$$

$$T_9 = shave(x_2)$$

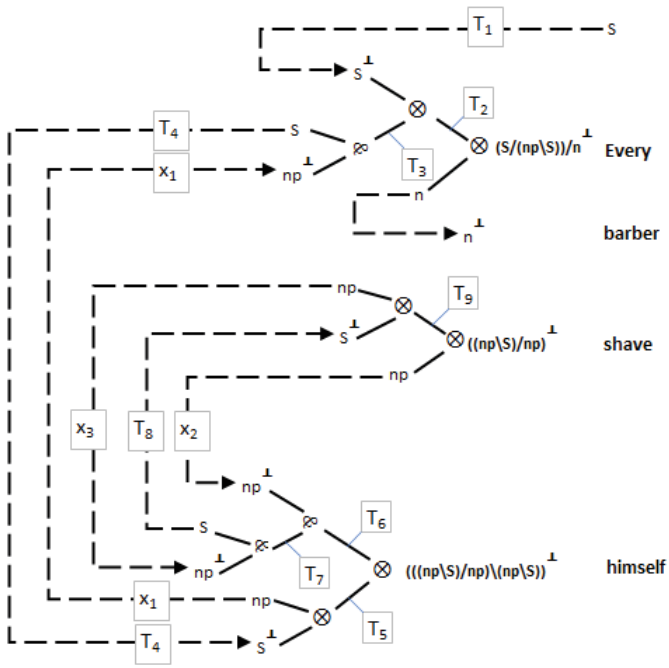


FIGURE 2.4: The categorial proof-net analysis with labels for the example "Every barber shaves himself."

By some straightforward substitutions, we would have the following syntactic analysis which is of type S:

$$T_1 = (every(barber))(\lambda x_1.(himself(\lambda x_3.\lambda x_2.(shave\ x_2)x_3))x_1)$$

Definition 2.10. We can define a morphism from syntactic types in Lambek Calculus (such as np, n and S) to semantic types: these semantic types are formulae defined from two types e (entities) and t (truth values or propositions) with the intuitionistic implication → as their only connective:

$$types ::= e \mid t \mid (types \rightarrow types)$$

Now, we can define a morphism from syntactic types to semantic types[4]:

(Syntactic type)*=	Semantic type	Description
S*	t	a sentence is a proposition
np*	e	a noun phrase is an entity
n*	$e \rightarrow t$	a noun is a subset of the set of entities
(a \ b)*= (b / a)*	a* → b*	extends (_)* to all syntactic types

Example 2.7. A common noun like *barber* or an intransitive verb like *sneeze* have the type $e \rightarrow t$ (the set of entities which are *barber* or who *sneeze*) a transitive verb like *shave* is a two place predicate of type $e \rightarrow e \rightarrow t$ (the pairs of entities such that the first one takes the second one) etc.

Example 2.8. Now, we can substitute the syntactic analysis that we gained in the example, i.e. T_1, with the following semantic counterparts and perform the alpha and beta conversion to gain the logical formula in a Montague-like approach:

word	**syntactic type u** **semantic type u*** **semantics: λ-term of type u***
every	$(S/(np\backslash S))/n$ $(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$ $\lambda P^{e\rightarrow t}\lambda Q^{e\rightarrow t}(\forall^{(e\rightarrow t)\rightarrow t}(\lambda x^e(\Rightarrow^{t\rightarrow(t\rightarrow t)}(Px)(Qx))))$
barber	n $e \rightarrow t$ $\lambda x^e(barber^{e\rightarrow t}x)$
shaves	n $e \rightarrow (e \rightarrow t)$ $\lambda x^e\lambda y^e((shaves^{e\rightarrow(e\rightarrow t)}x)y)$
himself	$((np\backslash S)/np)\backslash(np\backslash S)$ $e \rightarrow (e \rightarrow t)$ $\lambda P^{e\rightarrow(e\rightarrow t)}\lambda x^e((Px)x)$

Now we start to substitute the semantic *lambda*-terms with the variables in the T_1, step by step and perform the beta-conversion:

$$T_1 = (every(barber))(\lambda x_1.(himself(\lambda x_3.\lambda x_2.(shave\ x_2)x_3))x_1)$$

[4]Logical constants for the usual logical operations such as disjunction (∨) conjunction (∧) and implication (→) have the semantic type $t \rightarrow (t \rightarrow t)$. The logical quantifiers such as ∀ and ∃ have the semantic type $(e \rightarrow t) \rightarrow t$

$$\downarrow \beta$$

$$every(barber) = \lambda Q^{e\to t}(\forall^{(e\to t)\to t}(\lambda x^{e}(\Rightarrow^{t\to(t\to t)}\ (barber^{e\to t}x)(Qx))))$$

$$\downarrow \beta$$

$$\lambda x_1.(himself(\lambda x_3.\lambda x_2.(shave\ x_2)x_3))x_1) = \lambda x_1.((shave x_1)x_1)$$

Thus, finally we get:

$$\downarrow \beta$$

$$\forall^{(e\to t)\to t}(\lambda x^{e}(\Rightarrow^{t\to(t\to t)}\ (barber^{e\to t}x)((shave^{e\to(e\to t)}x)x)))$$

That can be re-written in an untyped and uncurrying form as below:

$$\forall x(barber(x) \Rightarrow shave(x,x))$$

2.3 Montagovian Generative Lexicon

Montagovian Generative Lexicon(=MGL) [Ret14] is a framework designed for computing the semantics of natural language sentences, expressed in many-sorted higher-order logic. The base types practiced in Montagovian-tradition (e and t) have fine-grained treatment in MGL since it has integrated many sorted types; and as a result, meaning are represented as many-sorted logic. This kind of representation provides a way to perform the restriction of selection. This framework is able to integrate a proper treatment of lexical phenomena into a Montagovian compositional semantics, like the (im)possible arguments of a predicate, and the adaptation of a word meaning to some contexts. MGL, uses many-sorted higher-order predicate calculus for semantic representation. It can be reified in first-order logic as well. For having sorts MGL uses classifiers in the language, this gives sorts a linguistically and cognitively motivated basis [MR13]; we have shown in detail how basic types and coerced terms can be gained by a serious game called *JeuxDeMots* as we will see in chapter 4. For meaning assembly, the second order λ-calculus (Girard system F) is used in order to factor operations that apply uniformly to a family of types.

There are many other good features in MGL such as ontological sub-typing that we have not provided detailed study here. We only focus on the main features of MGL and we will describe some technical parts of the framework such as coercions in the section 4 when we want to deal with the semantic gradience problem, and we discuss on Hilbert-epsilon operators in the chapter 3 when we want to deal with quantifier scoping and underspecification problems.

Many-sorted formulas in system F MGL uses a large number of base types and compound types. Though, it looks almost necessary to define operations over a family of similar terms with different types. This brings some flexibility in the typing and let us have terms that act upon families of terms and types. This is the philosophy behind using Girard's system F as the type system [Gir11]. System F involves quantified types whose terms can be specialized to any type. MGL can be written as ΛT_{y_n} since it extends T_{y_n} of Muskens [Mus90] with the second order operator and the corresponding quantified Π types.

The types of ΛT_{y_n} are defined as follows:

- Constant types e_i and t are (base) types.
- Type variables $\alpha, \beta, \ldots$ are types.
- Whenever T and α respectively are a type and a type-variable, the expression $\Pi\alpha.T$ is a type. The type variable may or may not occur in the type T.
- Whenever T_1 and T_2 are types, $T_1 \to T_2$ is a type as well.

The terms of ΛT_{y_n}, which encode proofs of quantified propositional intuitionistic logic, are defined as follows:

- A variable of type T i.e. $x : T$ or x^T is a term, and there are countably many variables of each type.
- In each type, there can be a countable set of constants of this type, and a constant of type T is a term of type T. Such constants are needed for logical operations and for the logical language (predicates, individuals, etc.).
- $(f\ t)$ is a term of type U whenever $t : T$ and $f : T \to U$.
- $\lambda x^T.t$ is a term of type $T \to U$ whenever $x : T$ and $t : U$.
- $t\{U\}$ is a term of type $T[U/\alpha]$ whenever $t : \Lambda\alpha.T$ and U is a type.
- $\Lambda\alpha.t$ is a term of type $\Pi\alpha.T$ whenever α is a type variable and $t : T$ is a term without any free occurrence of the type variable α in the type of a free variable of t.

The reduction of the terms in system F or its specialized version ΛT_{y_n} is defined by the two following reduction schemes that resemble each other:

- $(\lambda x^{\phi}.t)u^{\phi}$ reduces to $t[u/x]$ (usual β reduction).
- $(\Lambda\alpha.t)\{U\}$ reduces to $t[U/\alpha]$ (remember that α and U are types)

Girard has shown [Gir11] reduction is strongly normalizing and confluent every term of every type admits a unique normal form which is reached no matter how one proceeds. The normal forms, which can be asked to be η-long without loss of generality, can be characterized as follows:

Proposition (i): A normal Λ-term N of system F, β normal and η long to be precise, has the following structure:

$$\mathcal{N} = \overbrace{(\ \lambda {x_i}^{X_i} \mid \Lambda X_j\ )^*}^{\substack{\text{sequence of} \\ \lambda \text{ and } \Lambda \text{ abstractions}}} \overbrace{(\cdots(\ h^{(\Pi X_k | X_l \to)^* Z}}^{\substack{\text{head} \\ \text{variable}}} \overbrace{(\{W_k\} \mid {t_l}^{X_l})^*\ )\cdots)}^{\substack{\text{sequence of } \{\cdots\} \text{ and } (\cdots) \text{ applications} \\ \text{to types } W_k \text{ and normal terms } {t_l}^{X_l}}}$$

Example 2.9. This example illustrates how system F factors uniform behaviors. Given types α, β, two predicates $P^{\alpha \to t}$, $P^{\beta \to t}$, over entities of respective kinds α and β, for any ξ with two morphisms from ξ to α and β (see Figure 2.5), F contains a term that can coordinate the properties P ,Q of (the two images of) an entity of type ξ, every time we are in a situation to do so – i.e. when the lexicon provides the morphisms.

Definition 2.11. Polymorphic **AND** is defined as follow:

$$\&^{\Pi} = \Lambda\alpha.\Lambda\beta.\lambda P^{\alpha\to t}.\lambda Q^{\beta\to t}.\Lambda\xi.\lambda x^{\xi}.\lambda f^{\xi\to\alpha}.\lambda g^{\xi\to\beta}.(\&^{t\to t\to t}(P(fx))(Q(gx))).$$

Such a term is straightforwardly implemented in Haskell along the lines of Van Eijck and Unger [VEU10]:

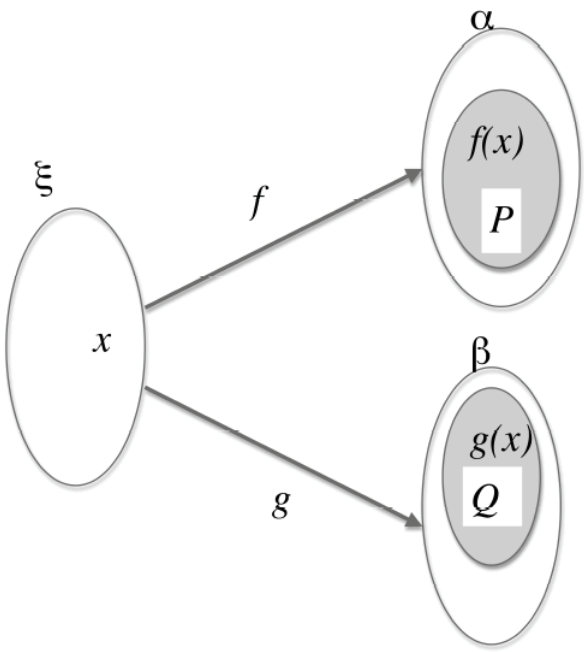


FIGURE 2.5: Polymorphic conjunction: $P(f(x))\&Q(g(x))$ with $x : \xi$, $f : \xi \to \alpha$, $g : \xi \to \beta$.

```
andPOLY :: (a -> Bool) -> (b -> Bool) -> c -> (c -> a) -> (c -> b) -> Bool

andPOLY =  p q x f g -> p (f x) && q (g x)
```

This can give an explanation for a "book" to be "heavy" as a "physical object", and to be at the same time "interesting" as an "informational content". The rigid use of possible transformations, as we define, would stop the over-generating artificial expressions.

The notion of coercion

Coercion (or transformation) is a key concept in *MGL*. It is a semantic notion which is rather intuitive. Each predicate contains— in its argument structure— the number and type of arguments needed. If each argument is present with the correct type, then a λ–term is formed by application, like in classical Montague semantics. In addition, the theory licenses certain cases of application ($f^{\alpha\to\gamma}x^{\beta}$) with $\alpha \neq \beta$ via various type coercions. If $\beta < \alpha$, i.e. β is a subtype of α, then the application is valid and is called a subtype coercion. If α and β are disjunct, then, there are two general strategies: argument-driven coercion or predicate-driven coercion. In argument-driven coercion the type mismatch can be treated by introducing a coerced term $g^{\beta\to\alpha}$ as in $f^{\alpha\to\gamma}(g^{\beta\to\alpha}x^{\beta}) : \gamma$. In predicate-driven coercion the type mismatch can be treated by introducing a coerced term $h^{(\alpha\to\gamma)\to(\beta\to\gamma)}$ as in $((h^{(\alpha\to\gamma)\to(\beta\to\gamma)}f^{\alpha\to\gamma})x^{\beta}) : \gamma$. More detailed information with relative examples can be found in the subsections 4.3.2 and 4.3.3.

Organization of the lexicon

In MGL, the lexicon associates each word w with a principal $\lambda - term$ $[w]$ which basically is the Montague term discussed earlier, except that the types appearing in $[w]$ belong to a much richer type system. This would enable us to impose some selectional restriction. In addition to this principal term, there can be optional $\lambda - term$ also called modifiers or transformations to allow, in some cases, compositions that were initially ruled out by selectional restriction.

There are two ways to solve a type of conflict using those modifiers. Flexible modifiers can be used without any restriction. Rigid modifiers turn the type into another one which is incompatible with other types or senses. This is the case for identity, which is always a licit modifier, is also specified to be flexible or rigid. In the later rigid case, it means that the original sense is incompatible with any other sense, although two other senses may be compatible. Consequently, every modifier, i.e. optional $\lambda - term$ is declared, in the lexicon, to be either a rigid modifier, noted (r) or a flexible one, noted (f).

As we will see, this setup in our lexicon would lead to manage the copredications in the linguistics this would offer a better control of incorrect and correct copredications. One can think that some meaning transfer differs although the words have the same type. An example for this case in French is provided by the words *classe* and *promotion*, which both refer to groups of pupils. The first word *classe* (English: *class*) can be coerced into the room where the pupils are taught, (the *classroom*), while the second, promotion (English: *class* or *promotion*) cannot. Consequently, we, in general, prefer word-driven coercions, i.e. modifiers that are anchored in a word.

Example 2.10. Considering the following lexicon (taken from [Ret14]):

word	principal $\lambda - term$	optional $\lambda - term$	rigid/flexible
book	$\hat{B} : e \to t$	$Id_B : B \to B$	(F)
		$b_1 : B \to \phi$	(F)
		$b_2 : B \to I$	(F)
town	$\hat{T} : e \to t$	$Id_T : T \to T$	(F)
		$t_1 : T \to F$	(R)
		$t_2 : T \to P$	(F)
		$t_3 : T \to Pl$	(F)
Liverpool	Lpl^T	$Id_T : T \to T$	(F)
		$t_1 : T \to F$	(R)
		$t_2 : T \to P$	(F)
		$t_3 : T \to Pl$	(F)
spread out	$spreat_out : Pl \to t$		
voted	$voted : P \to t$		
won	$won : F \to t$		

where the base types are defined as follows:

B	book	T	town
ϕ	physical objects	Pl	place
I	information	P	people
		F	football team

We can analyze the following examples (taken from [Ret14]):

2.10a. Liverpool is spread out.

2.10b. Liverpool voted.

2.10c. Liverpool won.

2.10d. Liverpool is spread out and voted (last Sunday).

2.10e. ? Liverpool voted and won (last Sunday).

- The example 2.10a leads to a type mismatch $spread_out^{Pl\to t} Lpl^T$, since "spread out" applies to "places" (type Pl) and not to "towns" as "Liverpool". It is solved using the optional term $t_3^{T\to Pl}$ provided by the entry for "Liverpool", which turns a town (T) into a place (Pl) $spread_out^{Pl\to t}(t_3^{T\to Pl} Lpl^T)$ – a single optional term is used, the (F)/(R) difference is useless.

- The example 2.10b if treated as the previous one, using the appropriate optional terms would lead to $voted^{P\to t}(t_2^{T\to P} Lpl^T)$.

- The example 2.10c if treated as the previous one, using the appropriate optional terms would lead to $won^{F\to t}(t_1^{T\to F} Lpl^T)$.

- The example 2.10d the fact that "Liverpool" is "spread out" is derived as previously, and the fact that "Liverpool voted" is obtained from the transformation of the town into people, who can vote. The two can be conjoined by the polymorphic "and" defined above as the term $\&^{\Pi}$ because these transformations are flexible: one can use one and the other. We can make this precise using only the rules of second order typed lambda calculus. The syntax yields the predicate $(\&^{\Pi}(spreat_out)^{Pl\to t} voted^{P\to t})$ and consequently the type variables should be instantiated by $\alpha := Pl$ and $\beta := P$ and the exact term is $\&^{\Pi}\{Pl\}\{P\} spreat_out^{Pl\to t} voted^{P\to t}$ which reduces to:
  $\Lambda\xi.\lambda x^{\xi}.\lambda f^{\xi\to\alpha}.\lambda g^{\xi\to\beta}.(\&^{t\to t\to t}(spread_out(fx))(voted(gx)))$.
  Syntax says that this term is applied to "Liverpool". Consequently, the instantiation $\xi := T$ happens and the term corresponding to the sentence is, after some reduction steps, $\lambda f^{T\to Pl}.\lambda g^{T\to P}.(\&^{t\to t\to t}(spread_out(f\ Lpl^T))(voted(g\ Lpl^T)))$. Fortunately the optional $\lambda - terms\ t_2 : T \to P$ and $t_3 : T \to Pl$ are provided by the lexicon, and they can both be used, since none of them is rigid. Thus we obtain, as expected $(\&^{t\to t\to t}(spread_out(t_3^{T\to Pl}\ Lpl^T))(voted(t_2^{T\to P}\ Lpl^T)))$.

- The example 2.10e is rejected as expected. Indeed, the transformation of the town into a football club prevents any other transformation (even the identity) to be used in the polymorphic "and" that we defined above. We obtain the same term as above, with *won* instead of *spread_out*. The corresponding term is: $\lambda f^{T\to Pl}.\lambda g^{T\to P}.(\&^{t\to t\to t}(spread_out(f\ Lpl^T))(voted(g\ Lpl^T)))$ and the lexicon provides the two morphisms that would solve the type conflict, but the one turning the Town into its football club is rigid, i.e. we can solely use this one. Consequently, the sentence is semantically invalid.

2.4 JeuxdeMots : A Lexical-Semantic Network

JeuxdeMots[5] (=JDM) is a lexical semantic network that is built by means of online games, so-called Games With A Purpose (=GWAP), launched from 2007 [Laf07]. It is a large graph-based network, in constant evolution, containing more than 310,000

[5] http://www.jeuxdemots.org

terms connected by more than 6.5 million relations. We have explained some technical aspect of this network in alignment with the problem of semantic gradience in the chapter 4. In this section, we only introduce the main features of the JDM.

Structure of the Lexical Network

The structure of the lexical network used in JDM is composed of nodes and links between the nodes, as it was initially introduced at the end of 1960s [CQ69] used in the small worlds [GDV08], and recently clarified in some studies [Pol06]. A node of the network refers to a term, usually in its lemma form. The links between nodes are typed and are interpreted as a possible relation holding between the two terms. Some of these relations correspond to lexical functions, some of which have been made explicit [MCP+95]. JeuxDeMots is intended for ordinary users and no special linguistics knowledge of the users is required.

Definition 2.12. A lexical network is a graph structure composed of nodes (vertices) and links:

- A node is a 3-tuple : <name, type, weight>
- A link is a 4-tuple : <start-node, type, end-node, weight>

The **name** is a string denoting the term. The **node type** is an encoding referring to the information of the node. For instance a node can be a term or a Part of Speech. The **link type** refer to the relation considered. A **node weight** is interpreted as a value referring to the frequency of usage of the term. The **relation weight**, similarly, refers to the strength of the relation. The following figure shows an example of the organization of the nodes and their relations.

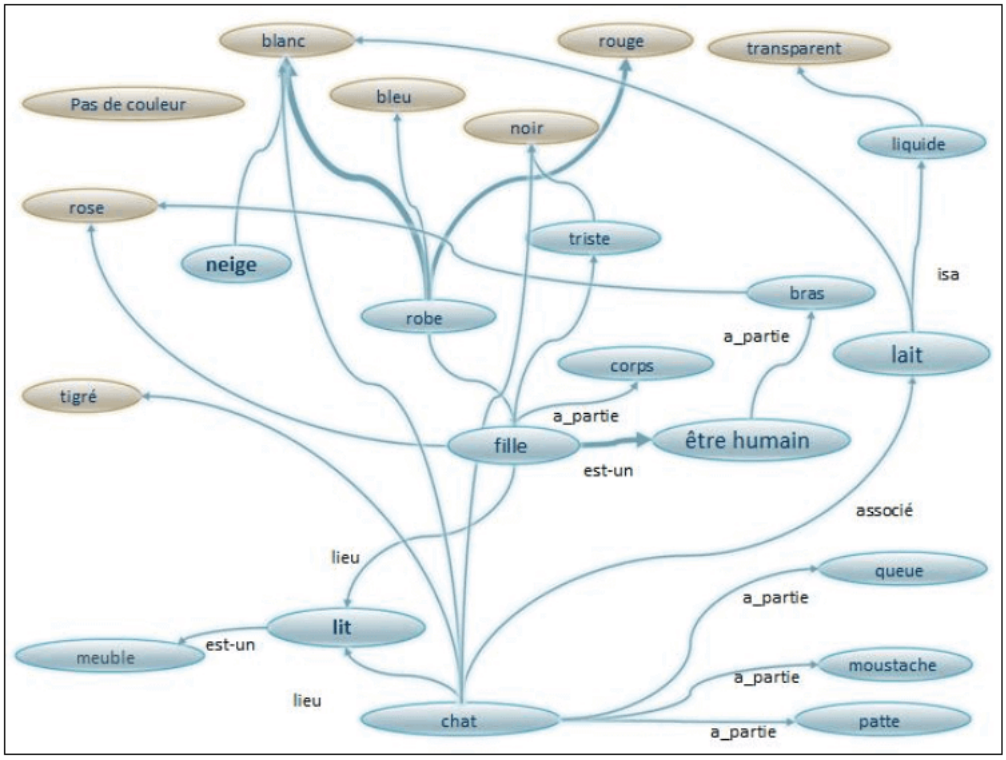


FIGURE 2.6: A part of the JDM network (taken form [LJ15]).

Definition 2.13. The link types are divided into following predetermined list of the categories:

- Lexical relations: such as synonymy, antonymy, expression, lexical family These types of relations are about vocabulary.
- Ontological relations: such as generic (hyperonymy), specific (hyponymy), part of (meronymy), whole of (holonymy).
- Associative relations: such as free association, associated feeling, subjective and global knowledge.
- Predicative relations: such as typical agent and typical patient. They are about types of relation associated with a verb and the values of its arguments (in a very wide sense) similar to (not identical) to semantic roles.

Validation of the relations in JeuxdeMots

The quality and consistency of the relations are essential for a lexical network. Due to this fact, the validation of the relations which are anonymously given by a player is made also anonymously by other players. A relation is considered valid if it is given by at least one pair of players. A game happens between two players. A first player, let's say player A, begins a game with prompting an instruction with a term T randomly picked from the database. The player A has then a limited time to answer which answer is applicable to the term T. The number of propositions which he can make is limited to letting the player not just type anything as fast as possible, but to have to choose the proper one. The same term with the same instruction is later proposed to another player B; the process is then identical. To increase the playful aspect, for any common answer in the propositions of both players, they receive a given number of points. At the end of a game, propositions made by the two players are shown, as well as the intersection between these terms and the number of points they win.

FIGURE 2.7: Screen-shot of an ongoing game with the target verb *fromage* (cheese) (taken form [Cha+17a]).

FIGURE 2.8: Screen-shot of the game result with the target noun *fromage* (taken from [Cha+17a]).

Figure 2.7 shows a screen-shot of the user interface while figure 2.8 shows a possible end of the game with the rewards. The figure illustrates answers of both players and the relevant scores.

Chapter 3

Modeling Meanings Preferences I: Ranking Quantifier Scoping

3.1 Introduction [1]

In this section, we start our preferential modeling on a particular linguistic phenomenon which is known as *quantifier scoping*.[2] The focus is more on the question of how to rank the valid logical meanings of a given multiple-quantifier sentence only by considering the syntactic quantifier order. Thus, we do not take into account the other possible factors such as common sense and lexical knowledge, and we only work on different semantic ambiguities resulted from the quantifiers scoping in the logical formula as our meaning representations.

The problem of quantifier scope ambiguity has first appeared in [Cho65] in the following classical examples:

Example 3.1.

3.1a. Everyone in the room knows at least two languages.
$\forall x(Person(x) \wedge In_the_Room(x) \rightarrow \exists y\ \exists z(Lang(y) \wedge Lang(z) \wedge y \neq z \wedge Know(x,y) \wedge Know(x,z))$

3.1b. At least two languages are known by everyone in the room.
$\exists y\ \exists z(Lang(y) \wedge Lang(z) \wedge y \neq z \wedge \forall x(Person(x) \wedge In_the_Room(x) \rightarrow (Know(x,y) \wedge Know(x,z)))$

Chomsky argues that the order of "quantifiers" in surface structures plays a role in semantic interpretation. As it can be observed in the relevant logical formula of the examples 3.1a and 3.1b are not synonymous. We can give 3.1a the reading where each person may speak a different two languages and 3.1b the reading where the same two languages are spoken by everyone. These two readings differ only in the scopes of the quantifiers. In 3.1a, the existential quantifier is inside the scope of the universal quantifier. Thus 3.1a could be true in a room where everyone spoke different languages and 3.1b would be false in that room since the existential quantifier is outside the scope of the universal quantifier. Although, 3.1b would only be true in a room where everyone speaks the same two languages.

VanLehn [Van78] argues that the most accessible scoped reading for sentences with n quantifiers Q_1 to Q_n is the one where the scopes are in order of mention,

[1] The material in this chapter is derived in large part from [CM17] which is the author's common work with Davide Catta. The research in subsections 3.2, 3.3, partially 3.4.2, 3.5 and 3.6 are done by the author himself.

[2] For a historical survey on this topic see chapter "The Natural History of Scope" in [Ste12].

with Q_1 widest and Q_n narrowest. The main factor that may override this preference is common-sense knowledge and this is something that we will not consider in our modeling in this chapter. The quantifier scope problem is not to describe all the factors which give these sentences their meanings. Because some of those factors involve discourse context and pragmatic knowledge and there is no precise formalization to capture those aspects.

Since sentences with quantifier scope ambiguities often have two (or more) non-equivalent semantic readings, capturing these phenomena with some plausible ranking techniques can potentially be useful for natural language inference systems. It is worth mentioning that different readings allow different inference paths and as it is shown in some studies that ([DGrt]) we can trace back the root cause of some *invalid inferences* in the misinterpretation of quantifiers as when occurs in quantifier scoping phase. Moreover, this kind of study is potentially suitable to be applied in some projects that gain meaning representations in logical forms from natural language texts by applying the Montagovian-style frameworks. One example of such project is e-fran AREN (ARgumentation Et Numerique) for the automated analysis of online debates by high-school pupils. The syntactic formalism that is used in the project is categorial grammars which are suitable for deriving meanings as the logical formulas. Grail [Moo17; Moo12], an automated theorem prover is used for parsing. It relies on a chart-based system close to categorial natural deduction which can rather smoothly be transformed to proof-nets.[3]

The problem of ranking the possible readings can be tackled if we employ the existing state-of-the-art techniques that use proof-nets for measuring the complexity of meaning representation [Mor00; Joh98]. We are specifically interested in Incomplete Dependency Complexity Profiling introduced in [Mor00], but as we will see adopting this technique to our problem is not as straightforward as it seems. Our goal in this chapter is to fix these issues and extend Incomplete Dependency Complexity Profiling technique for the representing and ranking semantic ambiguities.[4]

The rest of the chapter is organized as follows: section 3.2 summarizes Gibson's ideas on modeling the linguistic complexity of human sentence comprehension, namely IDT. In section 3.3, we recall the success and two limitations of IDT-based complexity profiling. In section 3.4, we define our new metric inspired by Hilbert's epsilon and reordering cost. We show how it fixes some problems in previous work and how it gives a correct account of new phenomena. In section 3.5, we discuss some limitations that our approach has. In the last section, we conclude and we explain possible future works.

3.2 Gibson's Incomplete Dependency Theory

Incomplete dependency theory is based on the idea of missing incomplete dependency. This hypothesis can be explained during the incremental processing of a

[3] Two important features of the AREN project: (i) since the data are provided in written language by the high-school pupils, we do not need to model intonation and prosody as a cue for disambiguation. (ii) there are, on average, 20 words in each sentence with almost two hundred category types, and the replies of the users happen with some delays. This restriction makes our solution scalable for this specific project that does not demand any big data analysis. Hence, it does not raise the issues of real-time processing and complexity.

[4] We will see in the chapter 6, how the introduced technique here can be combined with other approaches in linguistic complexity such as Distance Locality Complexity Profiling.

sentence when a new word attaches to the current linguistic structure. The main parameter in IDT is the number of incomplete dependencies from the new word to the existing structure. This gives an explanation for the increasing complexity of the examples 3.2a-3.2c. In 3.2a, *the reporter* has one incomplete dependency; in 3.2b, *the senator* has three incomplete dependencies; in 3.2c *John* has five incomplete dependencies at the point of processing. For the sake of space, we only explain the most complex case, i.e. 3.2a in which the incomplete dependencies at the moment of processing *John* are: (i) the NP *the reporter* is dependent on a verb to follow it; (ii) the NP *the senator* is dependent on a different verb to follow; and (iii) the pronoun *who* (before *the senator*) is dependent on a verb to follow; (iv) the NP *John* is dependent on another verb to follow; and (v) the pronoun *who* (before *John*) is dependent on a verb to follow. These are five unsaturated or incomplete or unresolved dependencies. IDT in its original form suggests calculating the maximum number of incomplete dependencies of the words in a sentence. One can observe that the complexity is proportional to the number of incomplete dependencies.[5]

Example 3.2.

3.2a. The reporter disliked the editor.

3.2b. The reporter [who the senator attacked] disliked the editor.

3.2c. The reporter [who the senator [who John met] attacked] disliked the editor].

3.3 Incomplete Dependency Complexity Profiling, and its limitations

An IDT-based proposal for measuring the linguistic complexity [Mor00] is based on the categorial proof nets. The general idea is simple: to re-interpret the axiom links as dependencies and to calculate the incomplete dependencies during the incremental processing by counting the incomplete axiom links for each word in a given sentence. This is almost the same as Gibson's idea in his IDT, except for the fact that he uses some principle of Chomsky Government-Binding theory [Cho82] instead of the categorial proof nets. The notion of counting incomplete dependency for each node, called complexity profiling, is more effective in terms of prediction than approaches that only measure a maximum number of the incomplete dependencies or the maximum cuts [Joh98].

3.3.1 Formal Definitions and Example

We can rewrite IDT-based complexity profiling [Mor00] by the following definitions:

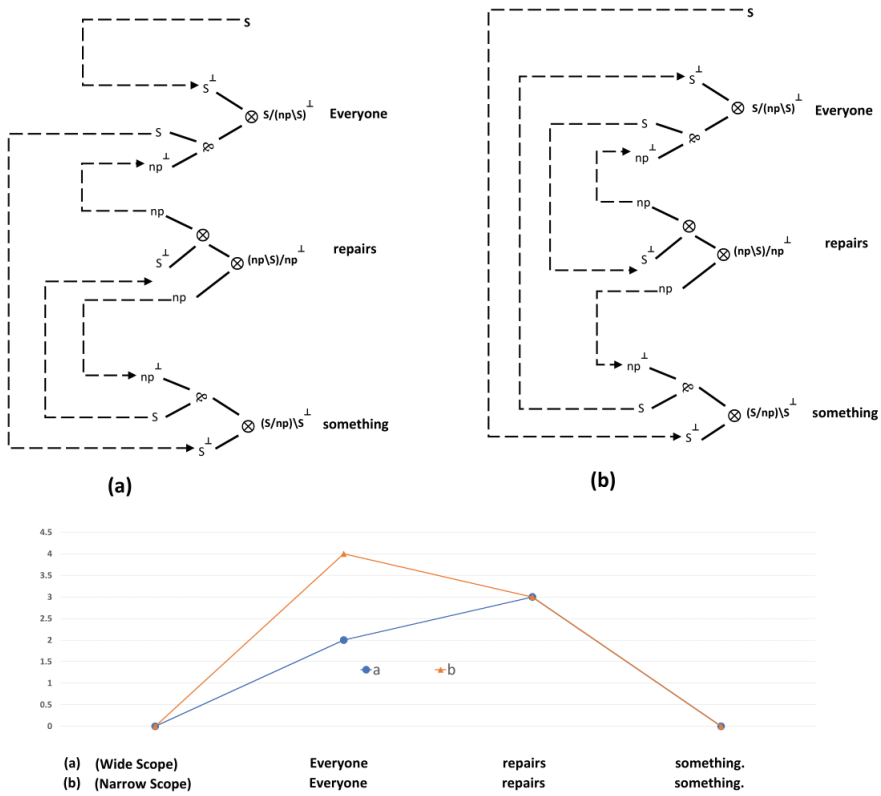
Definition 3.1. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i ($i \in [1, n]$). The incomplete dependency number of C_{i_0} in π, written as $ID_\pi(C_{i_0})$, is the count of axioms $c - c'$ in π such that $c \in (C_{i_0-m} \cup S)$ ($m \geq 0$) and $c' \in C_{i_0+n+1}$ ($n \geq 0$).

[5]We have calculated the linguistic complexity of the examples in this section in the chapter 5. To avoid unnecessary repetition, the detailed calculations of these specific examples are dropped in this section. The readers can refer to the chapter 5 and appendix B.

Definition 3.2. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^{\perp}, \ldots, (C_1)^{\perp}, S$. We define the IDT-based linguistic complexity of π, written $f_{idt}(\pi)$ by $f_{idt}(\pi) = (1 + \sum_{i=1}^{n} ID_{\pi}(c_i))^{-1}$.

Definition 3.3. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is IDT-preferred to π_j whenever $f_{idt}(\pi_i) > f_{idt}(\pi_j)$.

FIGURE 3.1: Proof net analyses for 3.3a (left hand) and 3.3b (right hand) with the relevant profiles.



We can show the two relevant proof nets in the figure 3.1 for the examples 3.3a and 3.3b. As it can be seen, the total sum of the complexity for 3.3b is greater than 3.3a, thus, it can predict correctly the preference of 3.3a over 3.3b which is what we expect. [6]

Example 3.3. Everyone repairs something.

3.3a. $\forall x \exists y Repair(x, y)$

3.3b. $\exists y \forall x Repair(x, y)$

[6]The readers who want to get more details on the calculation of the measurements can take a look at the relevant tables in appendix B which covers all the examples in this chapter.

As we can observe, the measurement uses syntactic analysis represented in the proof-net structure and gains the complexity profile by counting the number of unresolved dependencies of each word. This measurement is supposed to represent, in a numerical way, the course of memory load in optimal incremental processing; in other words, the metric of complexity profiles is motivated computationally as a measure of the incremental load on working memory of processing (unresolved) dependencies.

There are two general objections against the IDT-based complexity profiling approach; specifically, when it deals with the quantifier scope problem:

3.3.2 Objection I: A Minor Problem

The first objection, which is a minor problem, comes back to the choice of Lambek categorial grammar which simply can not derive all valid semantic readings. We address this objection in the section 3.4.2 by providing a complexity measuring technique that works only on the valid readings, and not over-generated readings that exists in the storage-based techniques [Coo83], [Kel88].

Example 3.4. Every researcher of a company saw a sample.

3.4a. $\forall x(Res(x) \wedge \exists y(Com(y) \wedge of(x,y)) \rightarrow \exists z(Sam(z) \wedge Saw(x,z)))$

3.4b. $\exists z(Sam(z) \wedge \forall x(Res(x) \wedge \exists y(Com(y) \wedge of(x,y) \rightarrow Saw(x,z)))$

3.4c. $\exists y(Com(y) \wedge \forall x(Res(x) \wedge of(x,y) \rightarrow \exists z(Sam(z) \wedge Saw(x,z)))$

3.4d. $\exists y(Com(y) \wedge \exists z(Sam(z) \wedge \forall x(Res(x) \wedge of(x,y)) \rightarrow Saw(x,z)))$

3.4e. $\exists z(Sam(z) \wedge \exists y(Com(y) \wedge \forall x(Res(x) \wedge of(x,y)) \rightarrow Saw(x,z)))$

3.4f. ?? $\forall x(Res(x) \wedge \exists z(Sam(z) \wedge of(x,y)) \rightarrow \exists y(Com(y) \wedge Saw(x,z)))$

One natural reaction to this problem is choosing proof-nets for Multiplicative Linear Logic(=MLL) instead of exploiting Lambek categorial proof-nets. We can consider the example 3.4 (taken from [HS87] with slight modification). Figures 3.2-3.6 show the categorial proof nets for all five valid readings. Some of them are constructed in the Multiplicative Linear Logic framework. Precisely speaking, proof nets for 3.4a and 3.4b which are illustrated in the figures 3.2 and 3.3, respectively are in Lambek categorial proof nets while the other three, namely 3.4-3.6 can only be constructed in the MLL proof-nets. With the following lexicon, and using the semantic reading that was introduced in the definition 2.9, we can gain all five valid logical formulas:

word	**syntactic type u** **semantic type u*** **semantics: λ-term of type u***
every	$(S/(np\backslash S))/n$ $(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\lambda Q^{e\to t}(\forall^{(e\to t)\to t}(\lambda x^e(\Rightarrow^{t\to(t\to t)}(Px)(Qx))))$
a	$((S/np)\backslash S)/n$ $(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\lambda Q^{e\to t}(\forall^{(e\to t)\to t}(\lambda x^e(\wedge^{t\to(t\to t)}(Px)(Qx))))$
researcher	n $e \to t$ $\lambda x^e(Res^{e\to t}x)$
company	n $e \to t$ $\lambda x^e(Com^{e\to t}x)$
sample	n $e \to t$ $\lambda x^e(Sam^{e\to t}x)$
saw	n $e \to (e \to t)$ $\lambda x^e\lambda y^e((Saw^{e\to(e\to t)}x)y)$
of	$(n\backslash n)/(S/(np\backslash S))$ $((e \to t) \to t) \to (e \to t) \to (e \to t)$ $\lambda P^{(e\to t)\to t}\lambda Q^{e\to t}\lambda x^e(\wedge^{t\to(t\to t)}(Qx)(P(\lambda y^e(of^{e\to(e\to t)}x)y)))$

Moreover, the only invalid reading, i.e. 3.4f, should be ruled out. It reads the sentence 3.4 as "expressing that for each representative there was a group of samples which he saw, and furthermore, for each sample he saw, there was a company he was a representative of" [HS87]. Thought, choosing MLL over Lambek would solve the problem for many cases, we have chosen to work on other existing algorithms for generating valid scoped quantifier readings, we will see this in great details in 3.4.2.

Now, it is time to look at a more important problem that IDT-based complexity profiling has.

3.3.3 Objection II: A Major Problem

The second objection, which is more important than the first one, is that IDT-based complexity profiling simply fails in some examples such as utterances that have sentence-modifier adverbials, nested sentences or direct speech. We can consider the following sentences:

Example 3.5. "Someone loves everyone", said John.

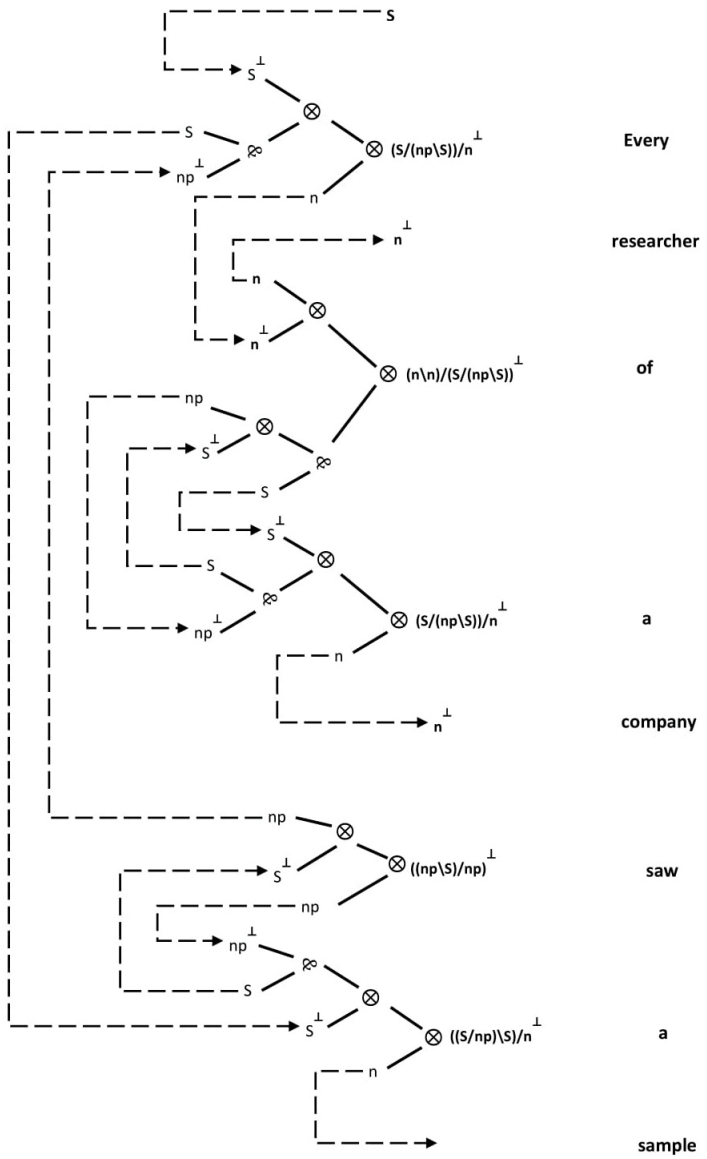


FIGURE 3.2: Proof net analyses for reading (a) of the example 3.4a.

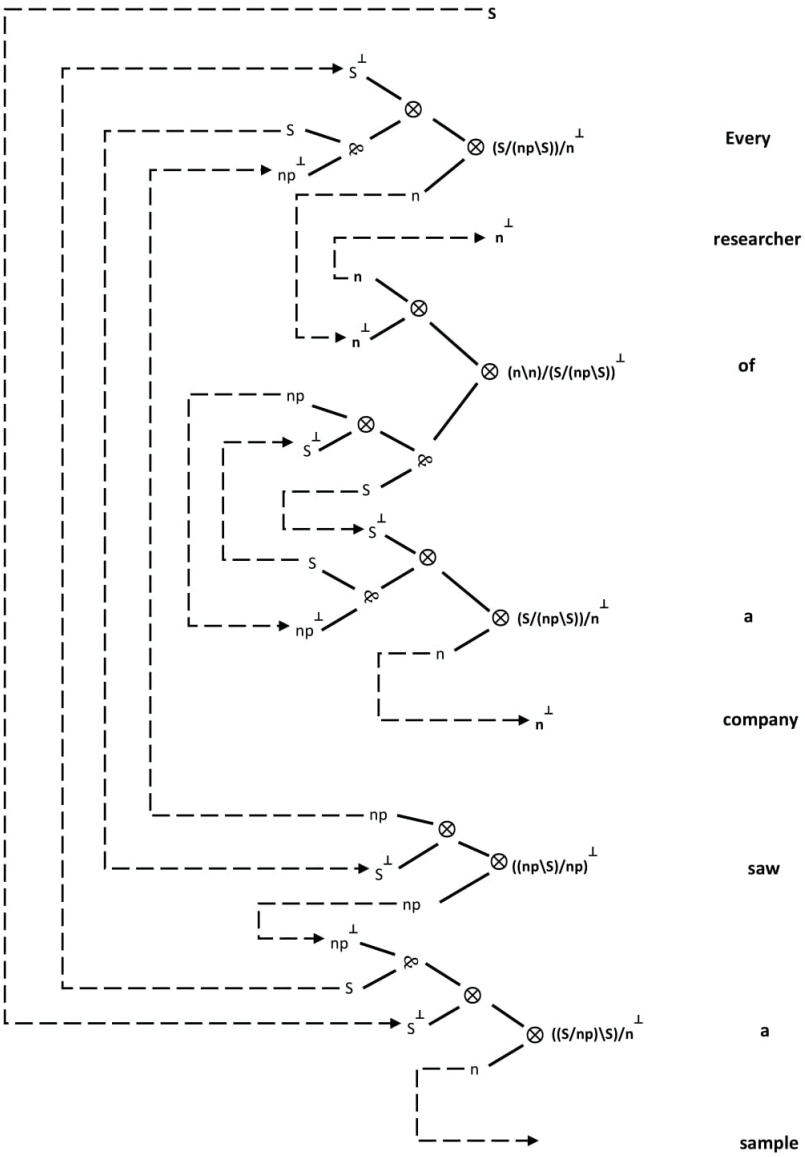


FIGURE 3.3: Proof net analyses for reading (b) of the example 3.4b.

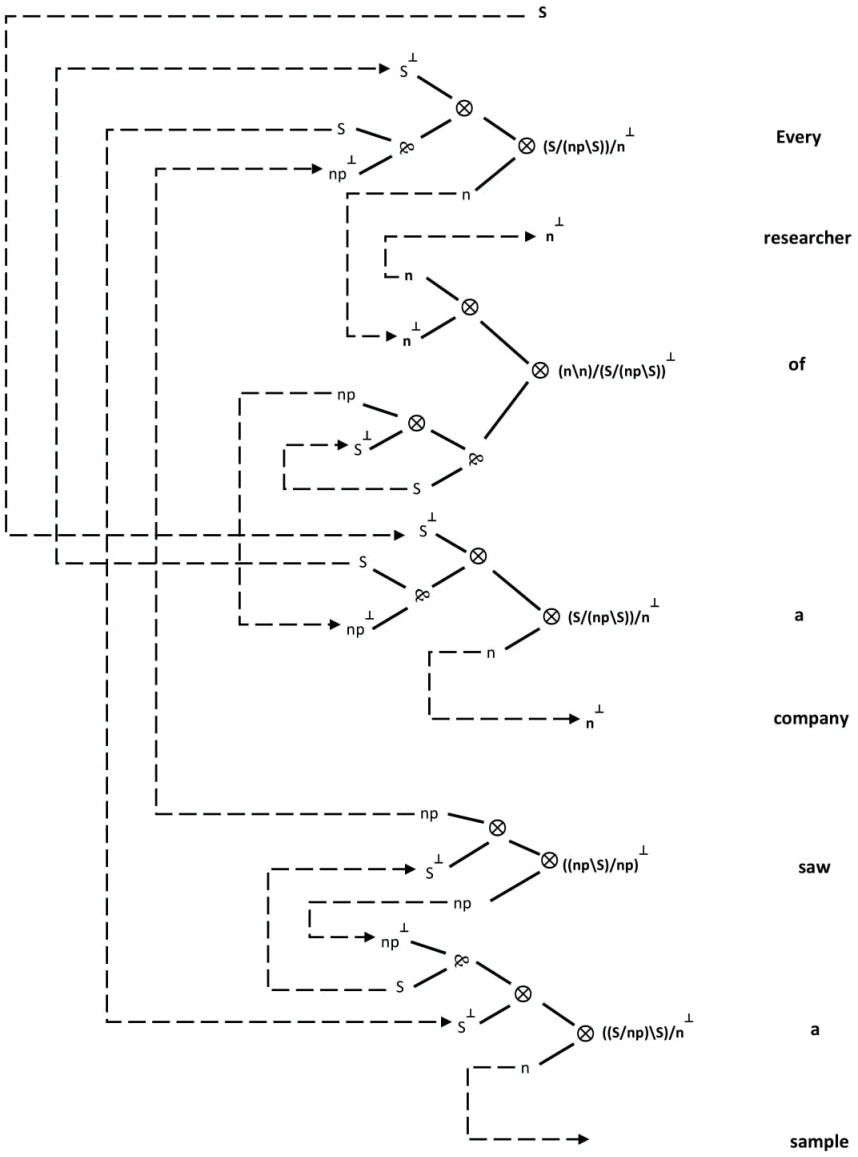


FIGURE 3.4: Proof net analyses for reading (c) of the example 3.4c.

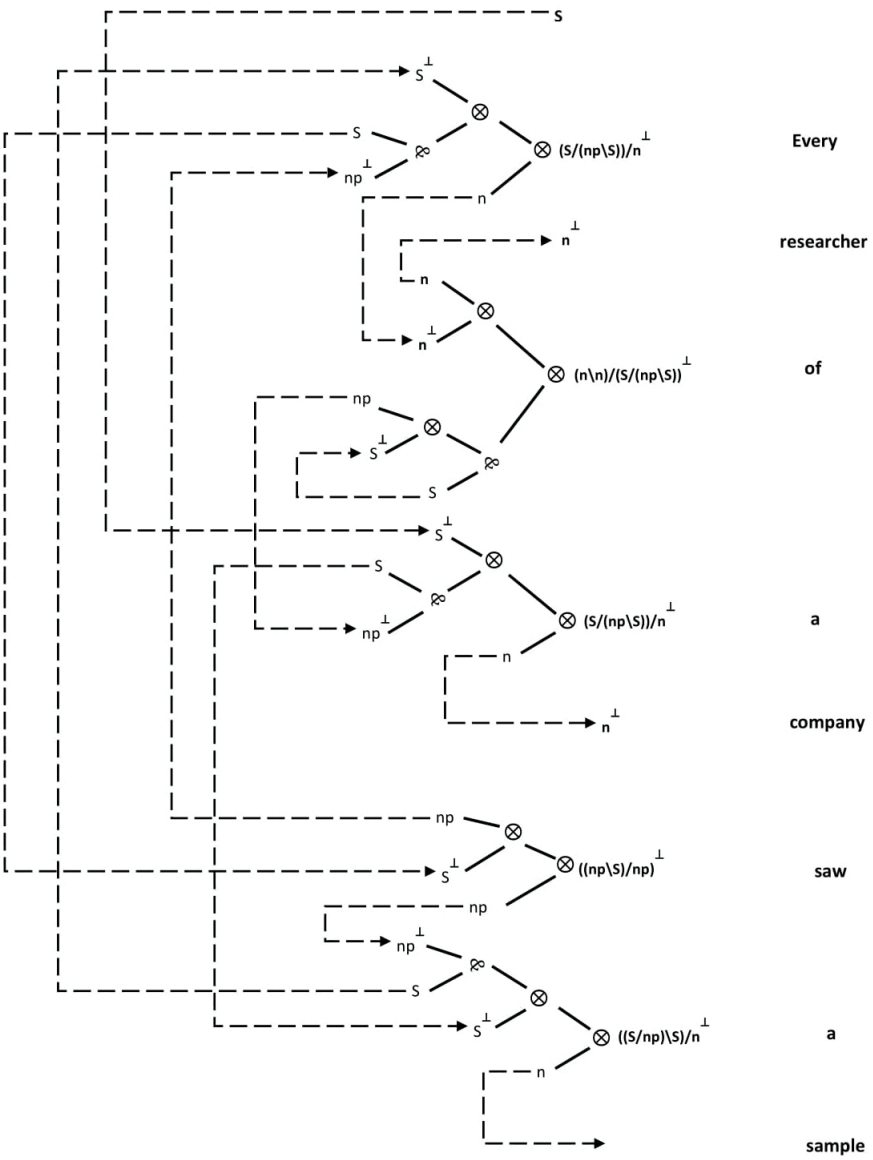


FIGURE 3.5: Proof net analyses for reading (d) of the example 3.4d.

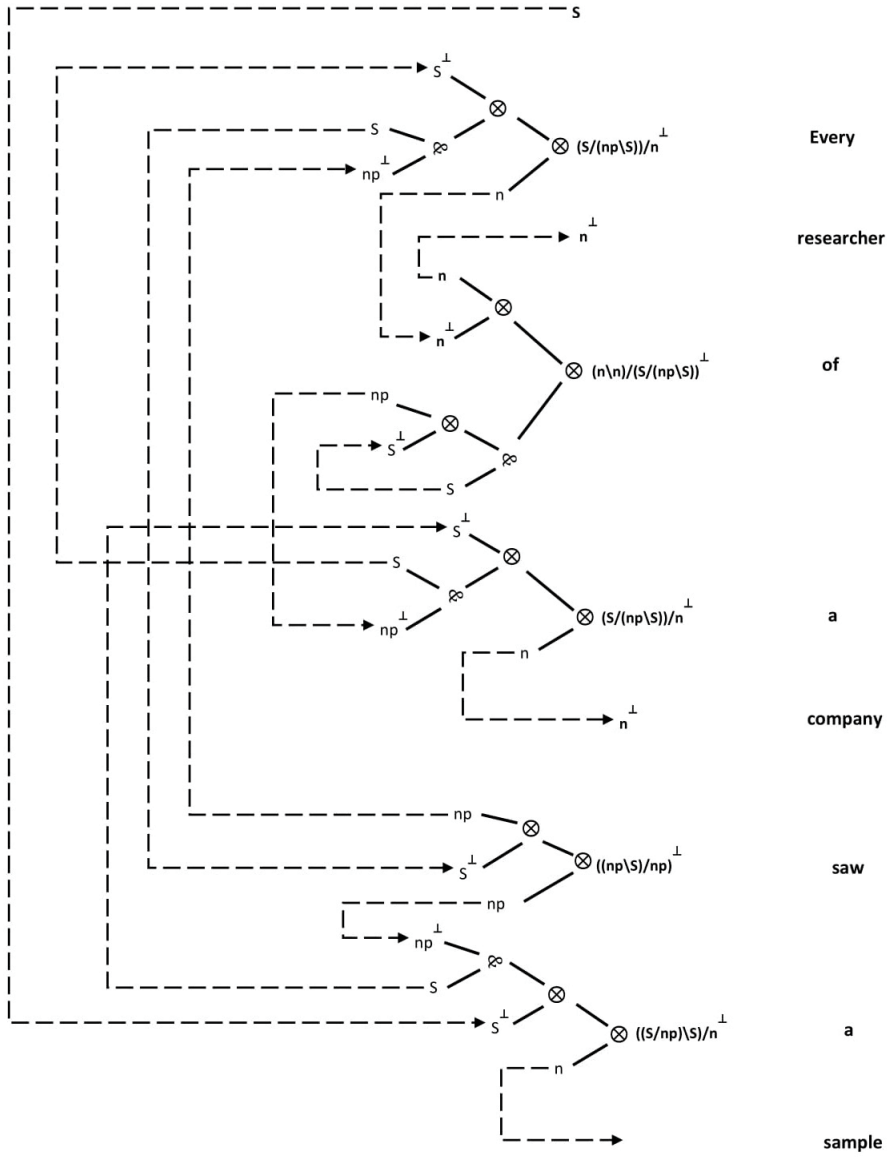


FIGURE 3.6: Proof net analyses for reading (e) of the example 3.4e.

Example 3.6. Everyone repairs something expertly.[7]

3.6a. $exp(\forall y \exists x Repair(x,y))$

3.6b. $exp(\exists y \forall x Repair(x,y))$

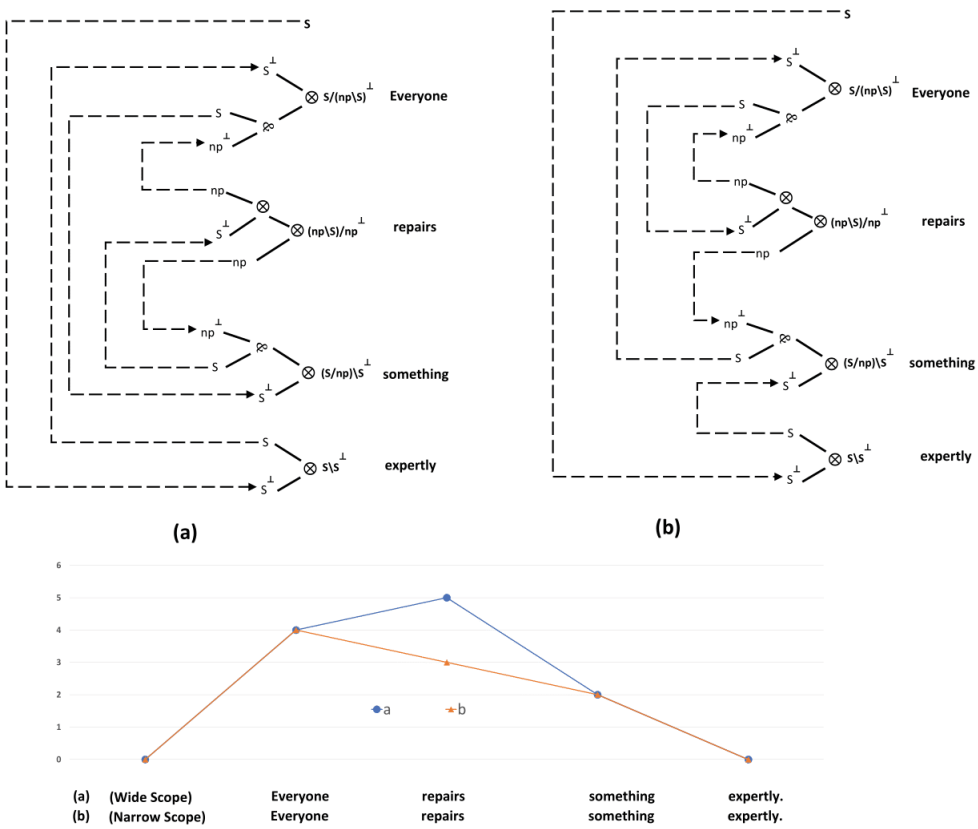


FIGURE 3.7: Proof net analyses for 3.6a (left hand) and 3.6b (right hand) with the relevant profiles.

Figure 3.7 illustrates two corresponding proof nets and complexity profiles for the example 3.6 [8]. As it is shown, the IDT-based complexity profiling technique for choosing the left-to-right preference fails, since the reading 3.6a which is supposed to have lower complexity is predicted wrongly. This objection which shows that IDT-based complexity profiling is context sensitive and it would naturally lead us to demand a new proposal for multiple-quantifier sentences.

[7] Thanks to one of the reviewers of our paper [CM17] that pointed up the distinction between VP-oriented adjunct and Clause-oriented adjunct as it is indicated in [HP+02, p.575-578]. Examples such as *someone loves everyone, unfortunately* belongs to a clause-oriented adjunct category. In this kind of examples, end positions for adverbs are strongly disfavoured unless there is a prosodic detachment. Our example in the paper is a kind of VP-oriented adjuncts and it prefers end position where prosodic detachment is not normal. Taken this grammatical note into account, we can stress that our example is correct without the need to use a comma before the adverb *expertly*.

[8] The proof net for the example 3.5 is not illustrated in this chapter. As it may be observed, it can be gained straightforwardly.

3.4 Hilbert's Epsilon, Reordering Cost and Proof-nets: A New Model

Although it is true that IDT-based complexity profiling fails in some cases related to the scoped readings, it is also true that it works rather well for other syntactic phenomena[9]. Considering this fact, our strategy is to keep IDT-based complexity profiling unchanged for those phenomena that work well and add a new procedure just for evaluating the complexity of quantification. We will see how we can filter out the complexity which can be raised by quantifiers. Formally, given two syntactic structures S_1, S_2 for a given sentence, the complexity can be evaluated in the following way:

$$C(S_1) > C(S_2) \iff (a_1, b_1) > (a_2, b_2)$$

where a_1, a_2 are the results of Morrill's criteria on S_1 and S_2, and b_1 and b_2 are the results of our proposal regarding quantifier order measurement (given in section 3.2). Notice that we are using lexicographical order, namely, $(a_1, b_1) > (a_2, b_2)$ iff $(a_1 > a_2)$ or $(a_1 = a_2$ and $b_1 > b_2)$. Broadly speaking, we will take the following steps:

1. Using in situ quantifier type assignment for construction of the categorial proof-nets in order to filter out the quantifier effects in complexity profiling phase. (Described in sub-section 3.1)
2. Measuring syntactic complexity profiles for the previous step.
3. Measuring quantifier distance for different valid readings. (Described in sub-section 3.2)
4. Introducing the preference relations from the last two steps.

3.4.1 In situ (=Overbinding) Quantification

Hilbert's ϵ-calculus ([Hil22]) is receiving a renewed interest. In particular, the application of ϵ-calculus to linguistics is becoming appreciated[10]. Our strategy for neutralizing quantifier effects in our complexity measurement on syntactic proof-nets can take place by exploiting Hilbert's ϵ-calculus in our semantic recipes.

In Hilbert's epsilon aside from the usual terms of first-order logic, we have the ϵ and τ terms. In particular if A is a formula and x is a variable then $\epsilon_x A$ and $\tau_x A$ are terms where all the occurrences of x in A are bound by ϵ and τ, respectively. What is interesting about these terms is that they express existential and universal quantification. In particular, $\epsilon_x A$ is the generic existential element and $\tau_x A$ is the generic universal element. The proper type for ϵ and τ is $(e \to t) \to e$. If we want to translate the sentence *All men are brave*, using the τ-binder, we can rewrite it as $brave(\tau_x man(x))$. An ϵ expression can take scope over the entire sentence even if its occurrence is nested in the parsed tree.

We can observe that the formula $ate(\epsilon\lambda y.(pizza\ y), \tau\lambda x.(child\ x))$ is a proper translation for the sentence *Every children ate a pizza* in Hilbert epsilon fashion. This

[9] We will see briefly in the chapter 5 which linguistic phenomena can be supported by IDT-based approach and what should we do for covering more linguistic phenomena that IDT-based approach cannot support.

[10] For an introductory explanation on Hilbert epsilon see [CPR17] and [Ret14, p.218-221].

formula does not correspond to any usual logical formula and it is very similar to underspecified representation. This underspecified representation which properly corresponds to syntax tree can help us to filter out the effect of quantifiers for the sentences in the complexity measurement phase of the categorial proof-nets. We do not have to perform the quantifier type raising in the syntactic phase in order to properly represent quantification. So, we can keep the syntactic type of quantifiers as being np/n. As discussed, typed Hilbert's epsilon [Ret14, p.218-221] suggests some properties that does not exist in other underspecified representations ([Cop+05], [Coo83] and [Ste99]). The first projection, namely a in our complexity pair representation (a, b), can be introduced by the following procedure:

1. Define np/n syntactic categories to all determiners and quantifiers in our lexicon.
2. Define proper Hilbert epsilon semantical representation for the quantifiers in our lexical recipes.
3. Construct the categorial proof-nets for the sentence with the categorial assignments in our lexicon and plug the lexical recipes into each word.
4. Calculate the complexity measurement profiles of all the valid constructed categorial proof-nets gained in the previous step. This is the first projection of the pairs in our preference semantic model.

3.4.2 Quantifiers Order Measurement

Now, we can introduce the second projection, namely b, in our complexity pair representation (a, b):

1. Given the logical formula which corresponds to the left-to-right reading of the sentence; add an index from 1 to n to each quantifier from left to right obtaining $Q_1, Q_2 \ldots, Q_{n-1}, Q_n F$ call this formula Φ.
2. By the procedure introduced in [HS87, p.49-53], derive all the valid quantifier (scope) readings of the sentence.
3. Let $\xi_1, \ldots, \xi_m$ be rewritten formulas obtained from the previous step.
4. Calculate for each ξ_i the penalty of quantifiers re-ordering as

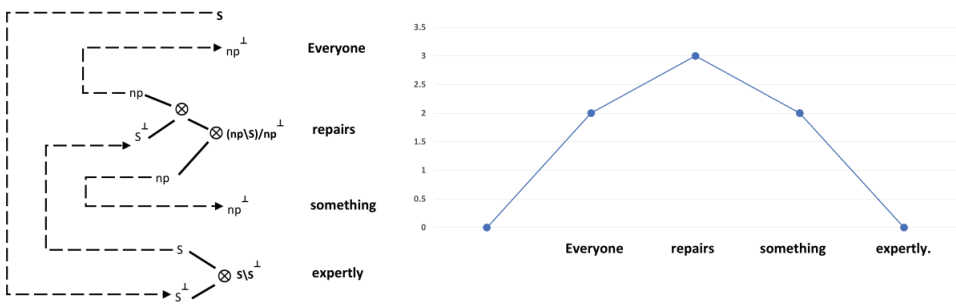
$$f_{qr}(\xi_i) = \frac{1}{\sum_{j=1}^{n} \mid j - Pos(Q_j, \xi_i) \mid +1}$$

   in which $Pos(Q_j, \xi_i)$ is the occurrence position of the quantifier Q_j in ξ_i counted from left to right and incremented from number one.[11]
5. Now, we have the preference measurement on the quantifier ordering of all valid logical readings as $f_{qr}(\xi_1), \ldots, f_{qr}(\xi_m)$.

[11] For development of the penalty function, we were inspired by preference calculation process based on the semantic distance in [Nag94].

3.4.3 Examples

FIGURE 3.8: New procedure for the example 3.6



Let us re-calculate the complexity profile of the example 3.6 with our new proposal. By the procedure provided in 3.4.1, we can have the proof-net as it is shown in the figure 3.8. By assigning the semantic recipes and having the associated λ-term to the proof-net we will have the unspecified semantic representation that naturally corresponds to the linguistic syntactic structure[12].

As the effect of quantifiers is neutralized we can see that the first projection of our pair is fixed by the number 7. So, we can calculate the second projection namely the score for each reading using the procedure in 3.4.2, thus we have:

Reading 3.6a is $\dfrac{1}{(|1-1|+|2-2|)+1} = 1$

Reading 3.6b is $\dfrac{1}{(|1-2|+|2-1|)+1} = \dfrac{1}{3}$

Thus, we have $(0.13, 1) > (0.13, 0.3)$, and this shows that reading 3.6a is preferred to reading 3.6b.

We can also perform the same procedure for the readings of the example 3.4:

Reading 3.4a is $\dfrac{1}{(|1-1|+|2-2|+|3-3|)+1} = 1$

Reading 3.4b is $\dfrac{1}{(|1-3|+|2-1|+|3-2|)+1} = \dfrac{1}{5}$

Reading 3.4c is $\dfrac{1}{(|1-2|+|2-1|+|3-3|)+1} = \dfrac{1}{3}$

Reading 3.4d is $\dfrac{1}{(|1-2|+|2-3|+|3-1|)+1} = \dfrac{1}{5}$

Reading 3.4e is $\dfrac{1}{(|1-3|+|2-2|+|3-1|)+1} = \dfrac{1}{5}$

[12] An anonymous reviewer has pointed out an objection: our proposed epsilon-style representation is not generally valid because of the equivalence $\tau_x Px \equiv \epsilon x \neg Px$, and the fact that the epsilon is intuitively interpreted as a choice function. This will affect us when non-P ($\neg P$) is chosen by the choice function. This objection will not cause any problem for our proposal since, firstly, the equivalence is not valid in intuitionistic logic; secondly, even if we use classical epsilon-calculus, it is debatable that $\epsilon - terms$ can be interpreted as choice functions. In fact, choice functions are only defined on non-empty domains. If we consider a domain of interpretation, where each element is in P, we can not interpret ϵ as a choice function given that there is no non-empty subset of $\neg P$ where the choice function can be defined. To our knowledge, there is no completely satisfactory model of the epsilon-calculus aside from the indexed version as in [Lei17].

One can observe that 3.4a is preferred over 3.4c, and 3.4c is preferred over 3.4b, 3.4d and 3.4e.

All in all, the new proposal truly predicts the quantifier left-to-right scoping while the IDT-based complexity profiling could not do so.

3.5 Limitations

There are two limitations against our new proposal:

The first important limitation is that in our semantic preference modeling our modeling on the quantifiers order— which is based on the surface syntax —does not always give us the exact human preference interpretation. For example, the logical formula represented in the 3.7a is not the meaning that is really preferred by the human, although, the existential quantifier has the wider scope. The preferred meaning is 3.7b since our common sense knowledge tells us that each door-step of a museum needs a unique guard. This is a linguistic phenomenon that is really hard to be computationally modeled and we should admit that have no solution for the time being for this kind of the problems.

Example 3.7. A guard stands in front of each museum door-step.

3.7a. $\exists x(Guard(x) \wedge \forall y(Museum_Door_Step(y) \rightarrow Stand_In_Front_Of(x,z))$

3.7b. $\forall y(Museum_Door_Step(y) \rightarrow \exists x(Guard(x) \wedge Stand_In_Front_Of(x,z)))$

The second limitation is that we do not have any psycholinguistic evidence for the preference of 3.4c over 3.4b, 3.4d and 3.4e as it is illustrated in the example 3.4. We consider this a purely linguistic issue and for the time being, we are not aware of any theory for addressing this issue. Consequently, we can not be sure to what extent such kind of the preferences are plausible in terms of the human performance.

3.6 Conclusion and Possible Extensions

We have reported some problems in the IDT-based complexity profiling approach for measuring quantifier scoping preferences. We did that by introducing some linguistic phenomenon that can not be supported by IDT-based complexity profiling. We showed how exploiting Hilbert epsilon can neutralize the quantifier effect and let us introduce a new procedure for measuring the complexity of quantifier scoping. We can extend our on-going project in the following directions: (i) ideally, there should be a data-set annotated for human-preferred readings of various naturally occurring utterances. In the absence of this, we are going to create a test suite illustrating the various phenomena that interact with quantifier scope preferences, and then to evaluate and extend it with the proposed method over that test-suite. This experiment would pave the way for the extension of our model by integrating other aspects such as common-sense knowledge and lexical semantics. (ii) we can exploit Montagovian Generative Lexicon framework [Ret14] that uses multi-sorted logical formulas. This choice would let us apply semantic distance approaches [Nag94] by using lexico-semantic networks such as Jeux-De-Mot [Laf07; Cha+15].

Chapter 4

Modeling Meanings Preferences II: Lexical Meaning and Semantic Gradience

4.1 Introduction [1]

There are some studies [Pro10] that work on the robust computational model that accords with intermediate levels of acceptability in human syntactical judgments or what is technically called as *the modeling human assessments of syntactic gradience*[Pro08]. In this chapter, we will focus on more complex linguistic gradients which go far beyond the syntactic level of sentence comprehension. We will work on the *semantic gradiences* that sometimes happen in sentences due to the implicit coercions in the linguistic meaning. This linguistic phenomenon can be illustrated by the following examples introduced in the *Generative Lexicon* framework [Pus91; Pus95]:

Example 4.1.

4.1a. Mary began the book.

4.1b. ?John began the dictionary.

4.1c. ??Mary began the mountain.

Example 4.2.

4.2a. Une femme a fini le livre de Villani. (A woman finished Villani's book.)

4.2b. ?Une chèvre a fini le livre de Villani. (?A goat finished Villani's book.)

Examples (4.1a-4.1c), taken from [Pus95] with slight modification, shows the degrees of semanticality in rather significant ways. As stated in [Pus91; Pus95] the sentence in (4.1a) admits of two strong interpretations, i.e. doing what one normally does to a book as a reader, reading; and doing what one normally does to a book as a writer, writing. There might be other possibilities for interpreting (4.1a) but this does not effect acceptance of (4.1a) in human comprehension performance. Example (4.1b) is less acceptable comparing to (4.1a). This is basically because no one is interested in reading a dictionary in a sense that people read a book. Of course, there are some exceptions such as Malcolm X in prison, but in this case, the dictionary is a form of narrative. The human acceptance of example (4.1c) is really low since the sentence has no meaning without providing the context. These examples show a

[1] The material in this chapter is derived in large part from [Laf+p2] which is the author's common work with Mathieu Lafourcade, Bruno Mery, Richard Moot, and Christian Retoré. The research in the section 4.5 which focuses on preference mechanism and some case studies are done by the author.

gradience of acceptability by human performance. Examples (4.2a-4.2b) shows the same phenomena in the French language. As one may guess, the meaning of (4.2a) is more acceptable comparing to (4.1a). Although two examples are syntactically well-formed, we can see a gradience of acceptability in these kinds of sentences. An interesting question is that how we can treat computationally this kind of semantic preferences in our model.

In order to be able to perform such a task, we need to have a rich lexical information to compute the meaning of utterances such as (4.2a-4.2b). Frameworks based on Generative Lexicon theories [Pus95], such as Montagovian Generative Lexicon [Ret14], can have a rich logical representation using a Montague-like compositional process.[2] In these approaches, the natural language utterance is analysed in its syntax and semantics layers as in classical compositional **Montague grammar**, producing a **logical form**; typing the terms with a **rich system of sorts** intended to capture **restrictions of selection**, using a **semantic lexicon**, produces some **typing mismatches** whenever polysemous terms are **linguistically coerced** to one of their facets. We will see in great details all these concepts.

A crucial issue for these systems is then to have sufficient lexical resources (as a rich lexicon incorporating types and coercions) to function. Our main task in this chapter is how to build such a lexicon for the Montagovian Generative Lexicon. We will do this task by using crowd-sourced lexical data that is gathered by a serious game which is called *JeuxDeMots*. The frequencies of the lexical occurrences— which is automatically gathered by the game players— would play a key role in our ranking mechanism. The whole preference outcome, following [FW83], would be based on a mechanism called preference-as-procedure with four-folded components, namely production, scoring, comparison, and selection.

The rest of the chapter is organized as follows: section 4.2 provides an overview of the lexicon requirement. We will explain why this is an important concern in favor of semantic gradience modeling. In section 4.3, the mechanism of gaining the lexical data from crowd-sourced serious games is briefly explained. This includes different aspects such as the lexemes, sorts, sub-types and coercions. Moreover, the methodology of the lexical transformations in a lexical-semantic network is explicated. In section 4.4, the integration process and ranking the transformation by means JDM is explained. In section 4.5, we describe two kinds of the preference modelings which is followed by a case study gained from the actual data in the *JeuxDeMots*. In the last section, we conclude and we explain possible future works.

4.2 The Lexicon Requirements

The **semantic analysis** of natural language is a process that should produce a complete and structural meaning representation of a given text (such as a logical formula or a Discourse Representation Structure) that makes explicit the entities referenced in the text as well as their relationships. This is used for word sense disambiguation, resolution of co-references, natural language inference and other complex tasks.

Rich lexical information is required to compute the meaning of utterances such as *I am going to the bank*, as *bank* is ambiguous between a geographic feature and a

[2]The methodology that we will describe is applicable to the other frameworks such as [Ash11], [Bek14], [Coo07] and [Luo11].

service building. Frameworks based on theories such as [Cru86] and [Pus95], including [Ash11], [Bek14], [Coo07], [Luo11] and [Ret14] are able to obtain a rich logical representation using a Montague-like compositional process that correctly interprets sentences such as *I am going to the bank; they blocked my account*. They not only interpret the types of the lexemes involved as indicating that *bank* is a service building, but also that *they* is a reference to a financial institution that is introduced in the first part of the sentence, and then coerced to a relevant human agent in the second.

Such frameworks require a rich corpus of lexical resources incorporating some degree of world knowledge encoded as complex types and lexical coercions; several of these frameworks benefit from software implementations, such as [CL14] and [Mer17]. In these approaches, the natural language utterance is analysed in its syntax and semantics layers as in classical compositional **Montague grammar**, producing a **logical form**; typing the terms with a **rich system of sorts** intended to capture **restrictions of selection**, using a **semantic lexicon**, produces some **typing mismatches** whenever polysemous terms are **linguistically coerced** to one of their facets.

In the framework we have proposed, the *Montagovian Generative Lexicon* (MGL), detailed in [Ret14], this is characterised by a **mismatched application**, such as a functional predicate P requiring an argument of type B being applied to an argument a of type A: $(P^{B\to\mathbf{t}}\ a^A)$. This is resolved using a **lexical transformation** that serves as the representation of the **linguistic coercion** taking place, and which should be provided by either P or a. There are two possibilities: *adapting the argument* with a transformation $f_1^{A\to B}$, the application becoming $(P\ (f_1\ a))$, and *adapting the predicate* with a transformation $f_2^{(B\to\mathbf{t})\to(A\to\mathbf{t})}$, the application becoming $((f_2\ P)\ a)$. Depending on the transformations that are provided by the functional terms P and a, one or the other adaptation occurs. The phrase *the dinner was delicious but took a long time* (adapted from a canonical example, see e.g. [Ash11]) can schematically be represented as

$$(\text{and}\ (\lambda x.\text{delicious}\ x)\ (\lambda x.\text{long}\ x))\ \text{dinner})$$

Within a many-sorted system, *take a long time* is restricted to events (entities of sort **evt**, or a subtype thereof), *delicious* is restricted to food (entities of sort F), and at least a transformation is needed in order to predicate on the two facets of *dinner*. MGL resolves this by having *dinner* as a term of type $\mathbf{evt}\to\mathbf{t}$, possessing a transformation $f_c^{(\mathbf{evt}\to\mathbf{t})\to(F\to\mathbf{t})}$ which is a function mapping the dinner event to the food that was served at the dinner in question.

A crucial issue for these systems is then to have sufficient lexical resources (as a rich lexicon incorporating types and coercions) to function. In order to be successful, MGL and other type-theoretic logical frameworks for lexical semantics require:

For the Syntax-Semantics Analysis: A suitable syntax-semantics analyzer which is able to provide a sufficiently structured output for a Montague-style analysis. MGL can make use of Grail (presented in [Moo17]) easily, yielding λ-DRT-based outputs based on Type-Logical Grammars. Grail operates on extensive, corpora-driven grammars for French.

A System of Sorts: Each word (lexeme) should be associated to a typed term in a type theory (or typed λ-calculus) where types are functionally and inductively

built from *base types*. Some theories use *common name* as base types, as discussed in [Luo12]. In our approach, the functional base types correspond to lexical **sorts** that capture the semantic notion of *restrictions of selection*. In this chapter, we use the word *sort* when referring to the lexical notion, and *base type* for the technical, λ-calculus notion necessary for computation. While the exact definition and scope of these lexical sorts is debated, this can be decided arbitrarily, as long as they provide a starting point that can be enriched and refined if untreated restrictions of selection become apparent; in this chapter, we use sorts defined by the semantic features of our resources, as discussed in Section 4.3.1.

A Set of Lexical Coercions: The core of MGL-based lexical semantics is the set of lexical transformations (corresponding to the *linguistic coercions* available for each *lexeme*), that allows co-composition to occur. Transformations might also be *constrained* in their use (some may be incompatible with others), and be dependent on (or made easier by) a specific *context*. The difficulties in building a wide-coverage lexical database for MGL lies in the acquisition of all such relevant transformations for all lexemes.

To sum up, MGL needs lexical entries in a format such as the following:

Lexeme	Logical Term	Type	Comments
dinner	$(\lambda x.\text{dinner } x)$	$\mathbf{evt} \to \mathbf{t}$	As in Montague semantics, nouns are predicates; **evt** is for *events*, **t** for *propositions*
with dinner	f_c	$(\mathbf{evt} \to \mathbf{t}) \to (F \to \mathbf{t})$	Transformation to "food that was served at dinner" (type $F \to \mathbf{t}$, base type F for the *foodstuff* sort)
to take a long time	$(\lambda x.\text{long } x)$	$\mathbf{evt} \to \mathbf{t}$	Predicate of events
to be delicious	$(\lambda x.\text{delicious } x)$	$F \to \mathbf{t}$	Predicate of food

(MGL encompasses higher-order composition mechanisms that will allow operators such as *the* and *and* to compute the correct predications.). The main goal of this chapter is to show how we can obtain the *lexical transformations* (such as f_c above) for our lexical entries.

4.3 Lexical Data Crowd-Sourced from Serious Games

While lexical networks have been developed for a long time (*WordNet*, defined in [Mil95], being the reference for English), there have been several recent efforts to build collaborative, crowd-sourced resources that can reflect the current uses and relations of words by language speakers. One approach, given in [Cha+13], is to engage as many people as possible in a "serious" game (or, more accurately, a "game with a purpose") in order to identify and co-validate lexical and relational information by having different competent speakers of the language competing to identify lexical meaning and relations between words.

These games include *JeuxDeMots*, described in [Laf07], which provides a lexical network that comprises more than 200 million relations between words (as strings

of characters). *JeuxDeMots* is actively developed and has proven remarkably robust. One advantage of this network over expert-produced and corpus-based resources is that a large amount of world knowledge has been added by the players: facts such as restrictions of selection (as in *cats can meow*) or ontological inclusion (as in *armchairs are chairs*) are explicitly produced by human players, while they are hard to get from other sources because of their "trivial" nature.

Another advantage of JDM is that it provides weights based on the frequency of words and phrases which is obtained from the available data played by users. Generally, there are two kinds of weights provided as negative and positive integers. The positive weights show the degree of confidence in a relation, while the negative weights, in contrast, indicate the degree of opposition between the nodes. For instance, *Autruche* (ostrich) and *déplacement aérien* (flight) have a relation called *r-agent-1* with a negative weight of value -65 in the *JeuxDeMots* database.This negative relation is gained by the users and emphasize on this fact that ostrich can not fly.

This network can be used as a relevant source of lexical data for type-theoretic frameworks, as discussed in [Cha+17a] for Modern Type Theories. While that publication has demonstrated the capacity in which the *types* for each lexeme can be extracted and derived (an MGL-compatible lexicon with a different set of lexical sorts can easily be produced), we want to focus on the extraction of the *lexical transformations* from such lexical networks.

We will be using the lexical network created by *JeuxDeMots*, amended by several related open, contributive resources to make a complete resource known as *Rezo*, together with Grail as a syntax-semantics analyzer, and MGL for lexical semantics. This forms a complete, coherent treatment process for French text.

4.3.1 Lexemes, Sorts and Sub-Types

JeuxDeMots and many other lexical networks operate upon character strings, while MGL differentiates between *contrastively ambiguous* homonyms. Words such as *bank* are considered as having (at least) two different entries in the Generative Lexicon tradition: one of the sorts *Financial Institution* and the other of sort *Geographical Feature*, that happen to have the same string representation. In *JeuxDeMots* where the character string is the basic unit, there are two ways to detect contrastive ambiguity: *Semantic Features* and *Refinements*. A *Semantic Feature* is similar to a sort in the lexicon (and a base functional type), as it reflects a broad category of things that the character string can denote; there might be several of such features associated to each string. *Refinements* are single-meaning facets for this string that have been crowd-sourced for the express purpose of resolving the contrastive ambiguity.

For example, in French, *un bar* is ambiguous and can denote at least three different things:

- a place where drinks are served (as in the English "bar", roughly synonymous with "public house" and "café", polysemous via metonymy with the furniture item sharing this name and function);
- a kind of fish (in English, "sea bass");
- and a pressure unit (1 bar = 100kPa).

This is denoted in the data from *JeuxDeMots* as having different refinements, as well as having several semantic features, including some that are incompatible with each other (here, "bar" has "location", " artifact" and "living being" as features). Extracting the initial information from *JeuxDeMots* is straightforward. *Common nouns* are logical predicates $P(_)$ of some type $\tau \rightarrow \mathbf{t}$, with τ a *sort*, subtype of entities **evt**. The initial sort is given as a salient semantic feature in *JeuxDeMots*.

These initial sorts can be refined later as needed for selection restrictions, which are also included in the lexical network. The network details possible patients and agents of predicates, for example. If the crowd-sourced data indicate that several incompatible semantic features are available for a single word, it simply means that we will have several distinct entries for several different lexemes that have the same string representation.

The typing for word taking nouns as arguments (such as adjectives, verbs...) are derived from a similar process and has been thoroughly explored in [Cha+17a]. Specific sorts are added as has been proposed for MGL in [MMR15a] : specific sorts can be introduced for nouns denoting *groups of a given sort* (such as *committee* being a predicate that denotes a group of people, $\mathbf{g}_P$) and *massive entities* such as *water* being a mass physical noun, of sort $\mathbf{m}_\varphi$). Several "operational" terms, such as the polymorphic conjunction *and* and determiners derived from Hilbert operators (discussed in [MMR15b]), are added by hand.

We can also derive a sub-typing mechanism. However, as discussed in [MR15], MGL restricts this mechanism (as well as the strict notion of "coercion among types") to *ontological inclusions*. In the system of sorts, this ontological hierarchy can be detected in data given from *JeuxDeMots* as denoting *hyperonymy* quite easily (such relations are pervasive in lexical networks). No other type-driven coercions are included in the system, contrary to other approaches — we think that this is a sensible restriction.

4.3.2 Lexical Transformations

From the definitions of the MGL system that have been presented before, the way to determine the lexical transformations never has been explicitly mentioned. By playing *JeuxDeMots*, players effectively create a database of **coercions between words** that we need to process according to the lexical entries and their typings.

As explained before, there are two ways to resolve a **type mismatch** in an application such as $(P^{B\rightarrow\mathbf{t}}\ a^A)$: adapting either **the functional argument** or **the predicate**.

4.3.3 Adapting the Argument

The most common adaptation is done on the argument, using a relevant transformation $f^{A\rightarrow B}$, yielding the correctly-typed $(P\ (f\ a))$. There are two **origins** possible for the transformation f: the transformation f is either provided by the lexical entry associated with the argument term a itself, or by lexical entry associated with the predicate term P.

Argument-driven transformations

Argument-driven transformations are extracted directly from entries revealing several meanings, that will be constrained by predication. For example, the expression *a book* is itself polysemous; an applied predicate such as *read*, *finish*, or *pack* will have strong typing constraints that will select one or several meanings (we will elaborate on this canonical example in Generative Lexicon framework. In MGL, the lexeme *book* (understood as the common noun associated to a literary concept versus the calendar-related verb) will have *a single sort*, R for readable object and be typed $R \rightarrow \mathbf{t}$. The same lexeme will be associated to several transformations:

$f_{object}^{(R\rightarrow\mathbf{t})\rightarrow(\varphi\rightarrow\mathbf{t})}$ that will be used in *heavy book*, $f_{read}^{(R\rightarrow\mathbf{t})\rightarrow(\mathbf{evt}\rightarrow\mathbf{t})}$ and $f_{write}^{(R\rightarrow\mathbf{t})\rightarrow(\mathbf{evt}\rightarrow\mathbf{t})}$ used in *to finish a book*, $f_{author}^{(R\rightarrow\mathbf{t})\rightarrow(P\rightarrow\mathbf{t})}$ used in *a monarchist book*.

In all of these cases, the transformation is *part of the lexical entry for the argument* and *constrained by the typing of the predicate*: *pro-monarchy* applies to people (and is associated to the *agentive quale* in GL tradition) and is of type $P \rightarrow \mathbf{t}$, *to read* is of type $R \rightarrow \mathbf{t}$, *to finish* of type $\mathbf{evt} \rightarrow \mathbf{t}$ (both are associated to the *telic quale*), and *to pack* of type $\varphi \rightarrow \mathbf{t}$ (associated to the *physical facet* of the complex object *book*).

In order to infer these transformations from lexical networks such as *JeuxDeMots*, we list the various possible *target types* for the predicates that are listed as *common patients* for this word. Then, for each target type, we select the compatible *associations* listed for the word, and generate a lexical transformation associated to this word, of a given typing, labeled with the association that is given in the lexical network. We retain the weight (frequency) of the association in order to filter out the more dubious transformation at a later stage (this is also used to rank the preferred interpretations if several are available).

For this example, listing all strings associated with *livre* (book; there are more than 3000 associations in the network, weighted and labelled), we will scan for *nouns denoting humans* and obtain *auteur* (author) and *écrivain* (writer), both with very strong relative weights; for *action verbs denoting non-instantaneous events* (that can said to *begin* or *end*) and obtain *lire* (to read) and *écrire* (to write), with a much stronger relative weight for the former. This process is detailed in Section 4.5.2. Even if some associations provided by the players in the network are dubious, filtering by type and syntactic properties yield exactly what is needed for the lexicon.

Predicate-driven transformations

Predicate-driven transformations are not (all) to be found in lexical networks or crowd-sourced data, as they characterize a predication made "in the spur of the moment". For example, the predicate *to read* is associated to a transformation $f_{written}^{(\varphi\rightarrow\mathbf{t})\rightarrow(R\rightarrow\mathbf{t})}$ that allows sentence such as *I read the wall* to be felicitous (and simply supposes that there is something written on said wall). Of predicate-driven transformations, some will be derived from lexical data (*grinding* is a common example, whenever *food* is associated to a word denoting a *living being*); others will need to be generated whenever the predication occurs, and be invalidated or not. The dynamic aspect of the crowd-sourced lexical network will integrate additional transformations as they are deemed pertinent, or fashionable, by its community; this is a strong argument in favor of such resources.

4.3.4 Adapting the Predicate

Transformations that change the type of a predicate are less common in MGL mechanisms. They are either provided by adverbs or other modifiers to a predicate, or intrinsic to the predicate. The former will be changing the target typing of a predicate. Examples of the latter include the polysemy of readings between collective and distributive among plural predicates detailed in [MMR15a]; these should be added manually for the lexicon.

4.3.5 Constraints and Relaxation

Transformations are associated to compatibility constraints that can be defined either as logical operators or as arbitrary functions that filter the possible combinations of transformations; these are intended to suppress or signal hazardous co-predications. Our interpretation of such phenomena is the following:

- *predicate-modifying* transformations are cumulative, as in *we all lifted the pianos, working in pairs* (our account will provide a predicate coercion, further constrained to a *covering* reading by the complement);
- *argument-driven* transformations are compatible with each other and not constrained, and give rise to *felicitous co-predications* such as *the dinner was delicious but took a long time* and *heavy yet interesting book*;
- *predicate-driven* transformations are exclusive: only one of them can be used on a given entity, exclusive of any other transformations and of the original term; this can be seen when making *infelicitous predications* such as **fast and delicious salmon* or **Liverpool won the match and voted Remain*.

As suggested by several studies including [Ret14], the latter constraint can be relaxed. Lexical predicate-driven transformations such as *grinding* (that we can get from crowd-sourced data) are compatible with other meanings, as long as there is a syntactic break between the two predications (as in *that salmon was fast; it is delicious*). There are ways to relax the constraints of application of non-lexicalised predicate-driven transformations as well, but these are syntax-, discourse- or pragmatic- dependent.

Detecting and validating constraints on co-predication, beyond the simple claim above, can also be a crowd-sourced task. *JeuxDeMots* itself is not suitable for this; however, a recent effort, *Ambiguss* (available at `https://ambiguss.calyxe.fr/` and discussed in [LB17]) is a good blueprint for crowd-sourced co-predication validation. *Ambiguss* is a database of sentences containing ambiguous (polysemous) terms that ask players to compete in determining the possible readings for those terms in context, and thus will be able to detect whenever co-predications are accepted or rejected when enough data will have been crowd-sourced. Such resources would also be useful for the evaluation of the performance of MGL on word-sense disambiguation tasks.

4.4 Integrating and Ranking Transformations

4.4.1 Adding Collected Transformations to the Lexicon

The semantic lexicon in MGL associates a set of *lexical transformations* (as optional λ-terms) to each lexeme. When the pertinent data has been collected from *JeuxDeMots*,

the coercion is transcribed as a transformation with a functional type $A \rightarrow \tau$ where A is the type of the source lexeme and τ is the expected typing, predicted during the collection process. The target concept serves as the name of the transformation. This data can be directly input and used in our prototype implementation of MGL given in [Mer17] (as the only necessary data are a source type, a target type and a name), and is added to the list of transformations of the current lexeme.

4.4.2 Scoring Interpretations

Compositional Lexical Semantics can produce the precise meaning of a polysemous term in context. However, there are cases where the immediate context is not sufficient to totally determine the sense used for a word, and a composed phrase can still be ambiguous: in the example above, *finir un livre*, the entity *livre* (book) is coerced to an event with a duration, but there are many suitable lexicalised events that can be used, and thus many lexical transformations that have a correct typing that can be used to resolve the type mismatch. MGL usually resolves this by producing **several** interpretations: one interpretation (a well-typed logical formula with a suitable transformation placed whenever necessary) per available coercion. The production of MGL is thus not a single logical representation, but a **collection of representations**.

Associating a preference score to different possible interpretations given by a compositional formalism is not so common (although this practice is pervasive in statistical or machine learning approaches to word-sense disambiguation); the more convincing works on this matter are mostly derived from Preference Semantics described, e. g., in [FW83]. An interesting extension is [Nag94], giving an implementation of the constraint-based preference scoring.

In order to provide a preference ranking, we exploit a strong advantage of extracting coercions from crowd-sourced data: the most common preferred coercions associated to a lexeme will be clearly represented in the lexical network (because *JeuxDeMots* counts the number of times each fact has been contributed or validated). Using the relative weights of different coercions, MGL will then be able to associate a *relative score* (similar to a probability measure) to each logical representation produced, allowing to rank the interpretations (while still being able to generate all possible meanings).
Our proposal for producing this **score** consists in a simple procedure. In order to analyze and rank interpretations for *finir un livre* (finish a book) :

1. Starting from textual data syntactically analysed using Categorial Grammars, we obtain a full logical representation, the terms of which are typed using a **many-sorted semantic lexicon** (in which event sorts *DurableEvent* and *AtomicEvent* are differentiated), we first identify the **adaptation** taking place in applications with type mismatches such as ($\text{finish}^{DurableEvent}$ $\text{a_book}^{Readable}$).

2. The lexicon then uses data available from *JeuxDeMots* in order to build the **set of possible coercions**: taking all **relations to the argument first** *livre* (book), we **select the ones with compatible types** with the typing of the predicate: verbs denoting a non-instant action. In this example, there are only *lire* (680) and *écrire* (210) in the first two hundred relations.

3. Finally, normalizing to a probability-like measure, this yields a **score between 1 and 0** for several interpretations ranked by order of probability. In this example, this means a score of 0.764 for *lire* and 0.236 for *écrire*.

4. This is validated by the crowd-sourced glosses for the complete phrasal expression that appears in that same order above.

In the case where **no suitable coercion is given by the argument**, a predicate-driven transformation can be used, as discussed in Section 4.3.3. Lexical transformations provided by the (functional) argument are always more specific and preferred to generic, predicate-driven transformations; thus the latter only occurs when there is no other choice, and there is only one interpretation (hence this case does not need any preference score).

Specific contexts can provide coercions that are not scored by this procedure, or that have different preferences. For instance, different contexts of enunciation will change the order of preferences in *finish a book*, for instance placing *writing* before *reading* in the context of the Paris Book Fair, or adding new possible coercions in contexts such as a bookbinding workshop.

4.4.3 Correcting the Lexicon using Different Sources

JeuxDeMots is a strong and rich resource for French as it is, encompassing at the moment more than 200 million relations between more than 2.7 million terms (words, names, and phrases). However, one of its more interesting characteristics for our purpose is that it is a living resource, constantly updated by crowd-sourcing (there are several players logged in and actively contributing at all times) and experts (selected members of the team constantly correct, update and add linguistic data and world knowledge). This means that after acquiring the initial lexicon used for a MGL system by converting relevant data from *JeuxDeMots* as outlined above, the same source can be used in order to continuously update and correct our lexicon. The presence of *phrases* in data from *JeuxDeMots* can also be used to evaluate and validate our compositional system. We can, for instance, derive the type and transformations for the lexical entry of *finir* (*to finish*), do the same thing for *livre* (*book*), and conclude from a MGL composition the meaning of *finir un livre* (*finish a book*: what is *finished* is the event of *reading* or *writing* a book); this is validated by the associated meanings of *finir un livre* that appears as a phrasal expression in *JeuxDeMots*.

This process can be systematized and automated, as *JeuxDeMots* can be enriched by asking the players the meaning of an expression that has not yet been analyzed (within reasonable limits). The meaning of words in context, as predicted by our compositional system, can also be checked by the players of the serious game *Ambiguss*. Thus, the evaluation of our system, as well as its correction and continuous improvement can be crowd-sourced.

4.5 Preference Mechanism for Quantifying Semantic Gradience

4.5.1 Preference-as-procedure v.s. Preference-as-restriction

Some studies, the same as ours, use multiple-stages for making the final preferences [FW83]. In this study, the preference is viewed as a procedure for assigning scores to competing for alternative representations and choosing the best one. This

is far beyond the naive practice of preference in semantic that simply uses the strategy of preferences-as-restriction. Now, we will consider the four key elements of preference-as-procedure as introduced in [FW83]:

1. **Production:** it produces all sentence readings whether or not they contain preference violations;
2. **Scoring:** readings are scored according to how many preference satisfactions they contain;
3. **Comparison:** whether or not an individual reading is accepted depends on a comparison with other readings;
4. **Selection:** the best reading (that is, the one with the most preference satisfaction) is taken, even if it contains preference violations

The way we adopt the above general criteria is as follows: in the production phase we make a query on the relations in the lexicon which represents the coercions we need; in the scoring phase we have the queries sorted by the weights in the relations, and finally, we select the top-ranked readings. We see now briefly this general idea in a running example.

4.5.2 Case Study

We will concentrate on some examples in order to demonstrate how the ideas that are described in previous sections can actually be achieved using *JeuxDeMots* by illustrating the translation procedure from *JeuxDeMots* to proper meaning representations in Montagovian Generative Lexicon (MGL) framework. The case study originates from the example in the influential book *Generative Lexicon*. Several publications, starting with [Las95], have remarked that *to finish a book* can be predicted to have different meanings depending on its *subject* (agent). An account of this phenomenon proposed in [IL15] is that meaning is constructed in three phases:

1. lexical meaning is extracted from the words involved;
2. compositional meaning is created by the combination of predicate and object, with different interpretations occurring and the most ranked being *reading*;
3. contextual (world-knowledge) meaning is combined by meaning associated with the subject and the interpretations can be re-ordered or filtered out.

We will attempt to provide an adequate prediction using the resources at our disposal, with the idea that the necessary pragmatic information or world knowledge is encoded in the lexicon (and is present in *JeuxDeMots* as well).

The Straightforward Case

Let us consider the phrase *To finish a book* in a situation where we have Claire, a researcher, reading a book by Villani:

Example 4.3. Claire a fini le livre de Villani. (Claire finished Villani's book.)

In order to analyse such a sentence, we search for a more generic phrase in *JeuxDeMots*: **(a)** *la femme a fini un livre* (the woman finished a book) for (4.3). Lexical coercions to actions such as *reading* or *writing* are excepted from *JeuxDeMots*. Moreover, *JeuxDeMots* also gives us the built-in weights relation that results in the expected ranking on the obtained lexical coercions.

Lexicon Organization in MGL and Meaning Representation

A detailed explanation on lexicon organisation in MGL is available in [Ret14, Sec 2.4]. In summary, the lexicon in MGL associates to each word w a principal λ-term which is Montague term with a much richer typed system and optional λ-terms known as *modifiers* or *transformations* modelling *lexical coercions*. The following sample lexicon is designed for the example (a):

Lexeme	Main λ-term	Optional λ-terms
livre (obj)	$\text{book}^{R\to\mathbf{t}}$	$f_{read}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}$ $f_{write}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}$
femme	$\text{woman}^{P\to\mathbf{t}}$	
fini	$\text{finish}^{\alpha\to(\mathbf{evt}\to\mathbf{t})}$	

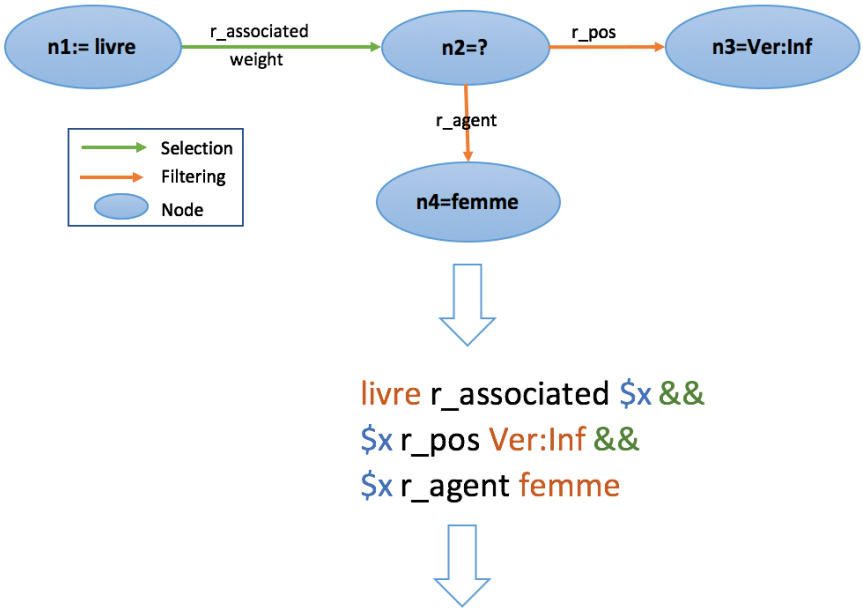
The sorts used here are as follows: P for *Person*, R for *Readable*, and **evt** for *Event*. Skipping unnecessary details on quantifiers and syntax, we can represent two possible meanings for (a) as
$((\mathit{finish}^{\alpha\to(\mathbf{evt}\to\mathbf{t})}\ (\text{the woman})^{P})\ (\text{a}\ (f_{read}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}\text{book}^{R\to\mathbf{t}}))^{\mathbf{evt}})$ and
$((\mathit{finish}^{\alpha\to(\mathbf{evt}\to\mathbf{t})}\ (\text{the woman})^{P})\ (\text{a}\ (f_{write}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}\text{book}^{R\to\mathbf{t}}))^{\mathbf{evt}})$.

Collecting Coercions

The relations gained from the game players in *JeuxDeMots* can technically be represented in different structures. The two kinds of data representation that are implemented and available are **relational** and **graph-based databases**. A simple query on an SQL system or on *Cypher*, the graph query language, on the *JeuxDeMots* graph-based database can fulfill our demand. We can also use *Datalog* which is a declarative database query language (basically Prolog with only constants and variables, i..e without terms, but with a proper negation). Although all of the options are technically available, we illustrate the process using a simple PHP-like query syntax which has actually been experimented.

As for example (a), what we want to do is basically to find the coercions *lire* (read) and *écrire* (write) for a sentence with the object *livre* (book) and the subject of sort *Person*. As illustrated in Fig. 4.1, we can find all the nodes with relation *r_associated* to the node *livre*. Two filters are then applied. The first one rules out the candidates that do not have the syntactic property of being a verb; to do so in *JeuxDeMots* we use *r_pos* relation that targets the node *Ver:Inf*. The second rules out the verbs that cannot have an agent of sort *Human*; to do so we use *r_agent* relation that targets the node *femme* (woman). We sort in descending order the final table with the built-in weight property of *r_associated* relation that exists in *JeuxDeMots*.

FIGURE 4.1: Obtaining Coercions: General Scheme, Query Code and Outcome Table for Example (a)



n1	weight	relation	n2	relation	n3	relation	n4
livre	680	r_associated	lire	r_pos	Ver:Inf	r_agent	femme
livre	210	r_associated	écrire	r_pos	Ver:Inf	r_agent	femme

The Non-Human Case

Regarding the extraction of the coercions from JeuxDeMots, we can see that the input of a query involves more than two words. For instance, considering the meaning of "dévorer un livre" (idiomatic French, *to devour a book* that can be used to denote binge-reading); in *JeuxDeMots* a relation between one of the meanings of "dévorer" and one of the meanings of "livre" depends on a third ingredient, namely the agent of "dévorer". Assuming that Blanche is Claire's pet goat, contrast the following:

Example 4.4.

4.4a. Claire a dévoré le livre de Villani. (Claire devoured Villani's book.)

4.4b. Blanche a dévoré le livre de Villani. (Blanche devoured Villani's book.)

4.4c. Claire a permis à Blanche de finir le livre de Villani. (Claire allowed Blanche to finish Villani's book.)

4.4d. Claire a promis à Blanche de finir le livre de Villani. (Claire promised Blanche to finish Villani's book.)

Observe that the last two examples with control verbs require a syntactic computation to determine the agent/subject of the action that is performed on the book. A further limit is that there are no sorts, types or sets in JeuxDeMots. If one is asked what a goat can eat, it is unlikely that a player answers *a book*. The answer: besides, grass, bushes, flowers, branches, leaves etc. a more generic answer would be "any object that is small and not too hard", but there is no single word corresponding to this class, while players answer words.

For this particular case, the fact that a goat may eat a book **is**, actually, included in JeuxDeMots. Some players included and validated the fact that a goat may eat *paper* and books being made of *paper* (this is indicated in the entry for *book* as a constitutive coercion), they can be eaten by goats. In this case, there even is also a direct fact that goats can eat books, but with a much weaker confidence.

Limits of MGL

In MGL as it stands now, coercions are attached to one word – except ontological inclusions which are encoded via sub-typing – but the actual coercion used to fix a type mismatch could be the combination of several coercions provided by all the words in the expressions. As observed above a coercion may be triggered by several words, and maybe the result of a sequence of relations. How can this be stored in MGL, in such a way that the general compositional mechanism of MGL produces the wanted readings and discards the unwanted ones? How can this be performed without trying all possible combinations of coercions, i. e., without going beyond reasonable time complexity limits?

A solution is to split the coercion into two (or more), each part being associated with each lexeme, the composition giving the complete result.

In that case, a type mismatch $P^{A\to\tau}(u^B)$ may be solved by first using a coercion $f^{B\to X}$ attached to u and a coercion $g^{X\to A}$ attached to P.

In all the above examples, there is a coercion from *books* to their physical facet. Having a *goat* as an agent will provide the verb *to eat* (which normally has an agent of sort *Animal* and a patient of sort *Food*) with a coercion from *Food* to physical objects, representing the *ingestion* of these objects and the world knowledge "fact" that "goats will eat (mostly) anything". That way, the goat Blanche may well eat Villani's book.

A Direct Solution

We expand our sample lexicon to this:

Lexeme	Syntax Constraint	Main λ-term	Optional λ-terms
livre (obj)	*subj*: P	$\text{book}^{R\to\mathbf{t}}$	$f_{read}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}$ $f_{write}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}$
livre (obj)	*subj*: A	$\text{book}^{R\to\mathbf{t}}$	$f_{eat}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}$
chèvre femme fini		$\text{goat}^{A\to\mathbf{t}}$ $\text{woman}^{P\to\mathbf{t}}$ $\text{finish}^{\alpha\to(\mathbf{evt}\to\mathbf{t})}$	

The syntactic constraint column extends the standard MGL lexicon in order to capture meanings that depend on the object and subject in a given sentence. In our case, having a subject of the sort either *Animal* or *Person* can significantly change the meaning and it obviously needs a different kind of coercions in the meaning representation layer. Considering the example:

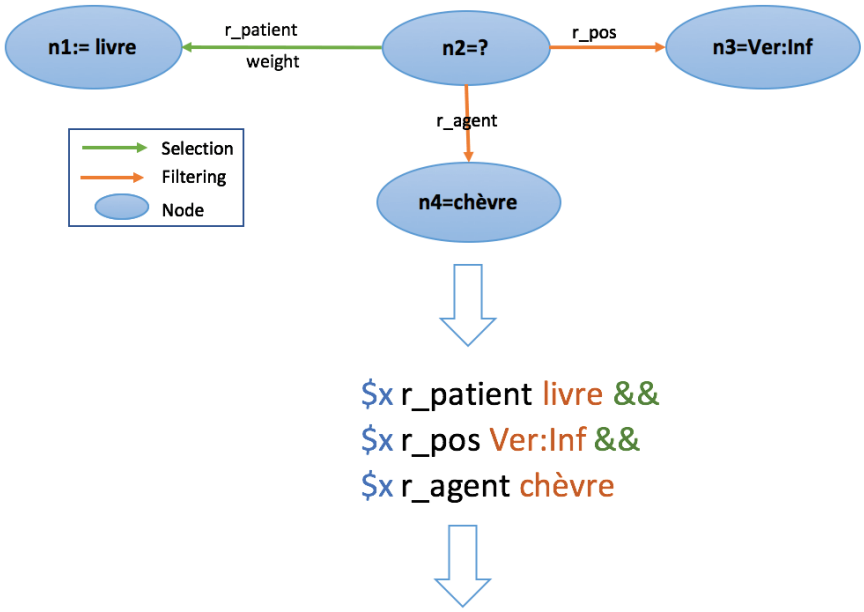
Example 4.5. Blanche a fini le livre de Villani. (Blanche finished Villani's book.)

We adapt this to a more generic sentence, **(b)** *la chèvre a fini un livre* (the goat finished a book) for (4.5), and the single reading for (b) should be

$((finish^{\alpha\to(\mathbf{evt}\to\mathbf{t})}\ (\text{the goat})^{A})\ (\text{a}\ (f_{eat}^{(R\to\mathbf{t})\to(\mathbf{evt}\to\mathbf{t})}\ \text{book}^{R\to\mathbf{t}}))^{\mathbf{evt}})$.

We want to find the coercion *manger* (eat) for a sentence with the object *livre* (book), the subject being the word *chèvre* (goat). As illustrated in Fig. 4.2, we can find all the nodes with relation *r_patient* to the node *livre*. As before, two filters are applied, selecting for verbs that can have *chèvre* as agent; we could then sort by weight if there were more than a single result.

FIGURE 4.2: Obtaining Coercions: General Scheme, Query Code and Outcome Table for Example (b)



n1	weight	relation	n2	relation	n3	relation	n4
livre	30	r_patient	manger	r_pos	Ver:Inf	r_agent	chèvre

This is direct relation extracted from *JeuxDeMots*; its confidence degree is light, and this is to be expected. In future work, we would like to acquire the coercion using "paper" as an intermediate step by searching for possible sequences (of limited length). Thus, we can use *JeuxDeMots* to derive the coercions that we need incorporate into the MGL lexicon.

4.6 Conclusion and Future Works

The Grail syntax-semantics analyzer, together with the type-theoretic account of lexical polysemy provided by MGL and based on compositional semantics and the ΛTY_n many-sorted logic, forms a computational system that is well-suited to lexical and semantic data crowd-sourced using *JeuxDeMots*. They can provide a complete chain of analysis that can process different complex linguistic phenomena for the French language. Grail has long been used in different versions, and has access to a large-covering French corpora-driven Type-Logical Grammar; *JeuxDeMots* provides access to a mature lexical network of words, phrases, and relations that is continuously updated, with publicly available and queryable data; we also have previously demonstrated the pertinence and computational applications of MGL. We have presented a process and experimental data that shows that this treatment chain works,

and can be automated. Thus, this system can be evaluated, corrected and updated in a semi-automated fashion using similar crowd-sourced data.

What remains to be done is mostly a work of integration of these various components. The possibility of modifying the existing MGL framework for allowing multi-part coercions to be added, as a composition of transformations licensed from different lexemes, should be examined in detail, as well as the implications of this modification to the time complexity of the computation, and the expressive power of the resulting formalism.

Chapter 5

Modeling Meanings Preferences III: Categorial Proof Nets and Linguistic Complexity

All things entail rising and falling timing.

Miyamoto Musashi

5.1 Introduction [1]

Linguistics and especially generative grammar à la Chomsky makes a distinction between *competence* and *performance* in the human processing of natural language [Cho65]. The competence is, roughly speaking, our ideal ability without the time and resource constraints to parse a sentence, i.e. to decide that it is grammatical or not. Competence is formally described by a formal grammar. The performance is how we actually parse a sentence; whether we succeed in achieving that by consuming some resources such as timing and memory; and how much the sentence resists to our attempt to analyze it. Computing the space and time algorithmic complexity is a fake solution because no one knows the algorithm being used by the human if it depends on the individual and on the kind of conversation. Even if it were so, nothing guarantees that space and time algorithmic complexity matches the degree of difficulty we experience when processing sentences. In this chapter, we try to provide a formal and computable account of the results of psycholinguistics experiences regarding linguistic complexity. We focus on syntactic complexity as studied in a number of linguistic processing phenomena such as garden paths, unacceptability of center embedding, preference for lower attachment, passive paraphrases acceptability and structures with embedded pronouns. The connection of our study in this chapter to the preference modeling is straightforward: The less complex linguistic structures are the more preferred ones from the point of view of linguistic human performance.

Regarding the psycholinguistics aspects, we mainly follow the studies by Gibson of linguistic complexity of human parsing. Gibson studied the notion of the nesting

[1] The material in this chapter is derived in large part from [MPR18] which is the author's common work with Jean-Philippe Prost and Christian Retoré.

linguistic difficulty [Gib91] in the examples 5.1a-5.1b. Gibson provided a computational psycholinguistic model that can predict that 5.1b is more complex than 5.1a. This came through with counting the maximal number of incomplete syntactic dependencies that the processor has to keep track of during the course of processing a sentence [described in the sec 5.2.1]. We refer to this theory as Incomplete Dependence Theory (IDT) as coined by Gibson.

Example 5.1.

5.1a. The reporter [who the senator attacked] disliked the editor.

5.1b. The reporter [who the senator [who John met] attacked] disliked the editor].

5.1c. The reporter [who the senator [who I met] attacked] disliked the editor].

IDT had some limitations for the referent-sensitive linguistic phenomena. To take an example, we can take a look at 5.1c which is similar to 5.1b except a replacement of the noun phrase *John* with the pronoun *I*. According to the discourse-based integration cost hypothesis, referents for the first-person pronoun *I* is already present in the current discourse, so, integrating across them consumes fewer cognitive resources than integrating across the new discourse referents *John*. This problem justified the later introduction of the Syntactic Prediction Locality Theory [Gib98]. A variant of this theory, namely Dependency Locality Theory (DLT), was introduced later by Gibson [Gib00] to overcome the limitations of IDT against the new linguistic performance phenomena [described in the sec 5.2.2]. In the original works, both IDT and DLT use properties of linguistic representations provided in Government-Binding Theory [Cho82].

On the formal side, in order to compute the complexity of a sentence — in a way that is inspired by Gibson's Distance Locality Theory — we use Lambek Categorial Grammar [Lam58] by means of proof nets construction [MR12b, Chap 6]. Proof nets were originally introduced by Girard [Gir87] as the mathematical structures of proof in linear logic. Categorial proof nets are to categorial grammar what parse trees are to phrase structure grammar. This kind of approach was initiated by Johnson [Joh98], who defines a measure of the instantaneous complexity when moving from a word to the next one (in particular for center embedded relative clauses) in a way that matches Gibson's and Thomas' analysis [GT96]. To define the complexity of a sentence, Johnson considers the maximum complexity between the words in a given sentence. This approach was refined by Morrill [Mor00], who re-interprets axiom links in categorial proof nets as incomplete (or unresolved) dependencies (sect 5.3). We rename this technique as *IDT-based complexity profiling* since it clearly inherits many aspects of Gibson's IDT, plus the new notion of profiling that exists in some psycholinguistic theories. This technique is quite successful at predicting linguistic performance phenomena such as garden paths, unacceptability of center embedding, preference for lower attachment and heavy noun phrase shift. Nevertheless, there is some predictive limitation for referent-sensitive phenomena such as structures with embedded pronouns. Our strategy to overcome this issue is to apply DLT instead of IDT on proof nets constructions which would lead to the introduction of *DLT-based complexity profiling*. We will show how this reformulation can improve the predictive power of the existing models in favor of the referent-sensitive linguistic phenomena.

The purpose of developing our computational psycholinguistic is not solely limited to measuring linguistic complexity. It is potentially applicable to some specific

tasks in the domain of the formal compositional semantics as we highlighted in the chapter 1. For instance, ranking different possible readings of a given ambiguous utterance, or more generally translating natural language sentences into weighted logical formulas.

The rest of the chapter is organized as follows: section 5.2 summarizes Gibson's ideas on modeling the linguistic complexity of human sentence comprehension, namely IDT and DLT. In section 5.3, we recall the success and limitation of IDT-based complexity profiling. In section 5.4, we define our DLT-inspired measure, we show how it fixes some problems in previous work and how it gives a correct account of those phenomena. In section 5.5, we provide more linguistic evidence to support our claim in favor of DLT-based complexity profiling over IDT-based approach. Section 5.6 explains some limitations that our approach potentially has. In the last section, we conclude and we explain possible future works. In particular, we compare the categorial grammar with other formalisms in terms of their usefulness for measuring linguistic complexity.

5.2 Gibson's Theories on Linguistic Complexity

We provide a very quick review of Gibson's IDT and DLT in order to make the readers familiar with their underlying concepts.[2] The question of how to automatically compute linguistic complexity based on both theories with categorial proof nets will be covered in sections 5.3 and 5.4.

5.2.1 Incomplete Dependency Theory

Incomplete dependency theory is based on the idea of missing incomplete dependency. The main parameter in IDT is the number of incomplete dependencies from the new word to the existing structure. The main parameter in IDT is the number of incomplete dependencies from the new word to the existing structure. This gives an explanation for the increasing complexity of the examples 5.1a-5.1c (repeated here as 5.2a-5.2c) which have nested relative clauses. In 5.2a, *the reporter* has one incomplete dependency; in 5.2b, *the senator* has three incomplete dependencies; in 5.2c *John* has five incomplete dependencies at the point of processing. For the sake of space, we only explain the most complex case, i.e. 5.2c in which the incomplete dependencies at the moment of processing *John* are: (i) the NP *the reporter* is dependent on a verb to follow it; (ii) the NP *the senator* is dependent on a different verb to follow; and (iii) the pronoun *who* (before *the senator*) is dependent on a verb to follow; (iv) the NP *John* is dependent on another verb to follow; and (v) the pronoun *who* (before *John*) is dependent on a verb to follow. These are five unsaturated or incomplete or unresolved dependencies. IDT in its original form suggests calculating the maximum number of incomplete dependencies of the words in a sentence. One can observe that the complexity is proportional to the number of incomplete dependencies.

Example 5.2.

5.2a. The reporter disliked the editor.

[2]We have explained IDT in the chapter 3. Nevertheless, for the convenience of readers, a quick review is provided.

5.2b. The reporter [who the senator attacked] disliked the editor.

5.2c. The reporter [who the senator [who John met] attacked] disliked the editor].

5.2d. The reporter [who the senator [who I met] attacked] disliked the editor].

5.2.2 Dependency Locality Theory

Dependency Locality Theory is a distance-based referent-sensitive linguistic complexity measurement put forward by Gibson to supersede IDT due to its predictive limitations. DLT posits two integration and storage costs. In this chapter, we have only focused on the integration cost. The complexity, interpreted as the locality-based cost of integration of two elements, depends on the intervened new discourse-referents. By performing a measurement on these referents, we can predict new linguistic phenomena, such as structures with embedded pronouns, illustrated in example 5.2d. The empirical experiences [WG99] support the acceptability of 5.2d over 5.2c. According to the discourse-based DLT structural integration cost hypothesis, referents for the first-person pronoun *I* is already present in the current discourse, so, integrating across them consumes fewer cognitive resources than integrating across the new discourse referents *John*. By means of just two aspects of DLT, namely the structural integration and the discourse processing cost we would be capable to predict a number of linguistic phenomena as we will see in details with a number of examples.

5.3 Incomplete Dependency-Based Complexity Profiling and its Limitation

An IDT-based proposal for measuring the linguistic complexity [Mor00] is based on the categorial proof nets. The general idea is simple: to re-interpret the axiom links as dependencies and to calculate the incomplete dependencies during the incremental processing by counting the incomplete axiom links for each word in a given sentence. This is almost the same as Gibson's idea in his IDT, except for the fact that he uses some principle of Chomsky Government-Binding theory [Cho82] instead of the categorial proof nets. The notion of counting incomplete dependency for each node, called complexity profiling, is more effective in terms of prediction than approaches that only measures the maximum number of the incomplete dependencies or the maximum cuts [Joh98].

5.3.1 Formal Definitions and Example

We can rewrite IDT-based complexity profiling [Mor00] by the following definitions:

Definition 5.1. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i ($i \in [1, n]$). The incomplete dependency number of C_{i_0} in π, written as $ID_\pi(C_{i_0})$, is the count of axioms $c - c'$ in π such that $c \in (C_{i_0-m} \cup S)$ ($m \geq 0$) and $c' \in C_{i_0+n+1}$ ($n \geq 0$).

Definition 5.2. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. We define the IDT-based linguistic complexity of π, written $f_{idt}(\pi)$ by $f_{idt}(\pi) = (1 +$

$\sum_{i=1}^{n} ID_{\pi}(c_i))^{-1}$.

Definition 5.3. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is IDT-preferred to π_j whenever $f_{idt}(\pi_i) > f_{idt}(\pi_j)$.

Example 5.3. We can show the two relevant proof nets for the example with subject-extracted relative clause, i.e. 5.3a in the figure 5.2; and the other example with subject-extracted relative clause, i.e. 5.3b in the figure 5.3. The relevant complexity profiles for 5.3a and 5.3b are illustrated in figure 5.1. As it can be seen, the total sum of the complexity for 5.3b is greater than 5.3a, thus, it can predict correctly the preference of 5.3a over 5.3b which is supported by measuring reading time experiments [GK98]. [3]

5.3a. The reporter who sent the photographer to the editor hoped for a good story.

5.3b. The reporter who the photographer sent to the editor hoped for a good story.

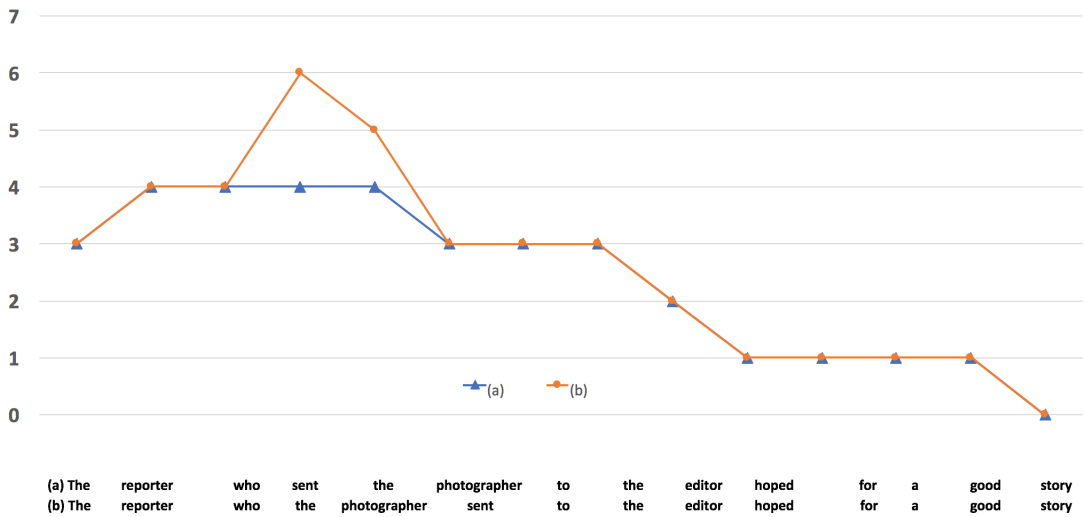


FIGURE 5.1: IDT-based Complexity Profiles for 5.3a and 5.3b.

5.3.2 Limitation

Obviously, the IDT-based account does not use DLT as its underlying theory. Not surprisingly, the linguistic phenomena that can only be supported by DLT would not be supported by IDT-based complexity profiling. Figures 5.5 and 5.7 shows this failure by constructing proof nets for 5.2c and 5.2d, respectively. As one may notice, the corresponding proof nets for the examples 5.2c and 5.2d are almost the same.[4] Consequently, IDT-based complexity profiling cannot discriminate both examples, i.e. it generates the same number for both sentences in contrast to the experiments [WG99]. This shows the importance of introducing DLT-based complexity profiling for proof nets in order to make more predictive coverage as we will do so.

[3] The readers who want to get more details on the calculation of the measurements can take a look at the relevant tables in appendix B which covers all the examples in this chapter.

[4] Following Lambek [Lam58], we have assigned the category $S/(np\backslash S)$ to pronoun I. Note that even assigning np, which is not a type-shifted category, would not change our numeric analysis at all.

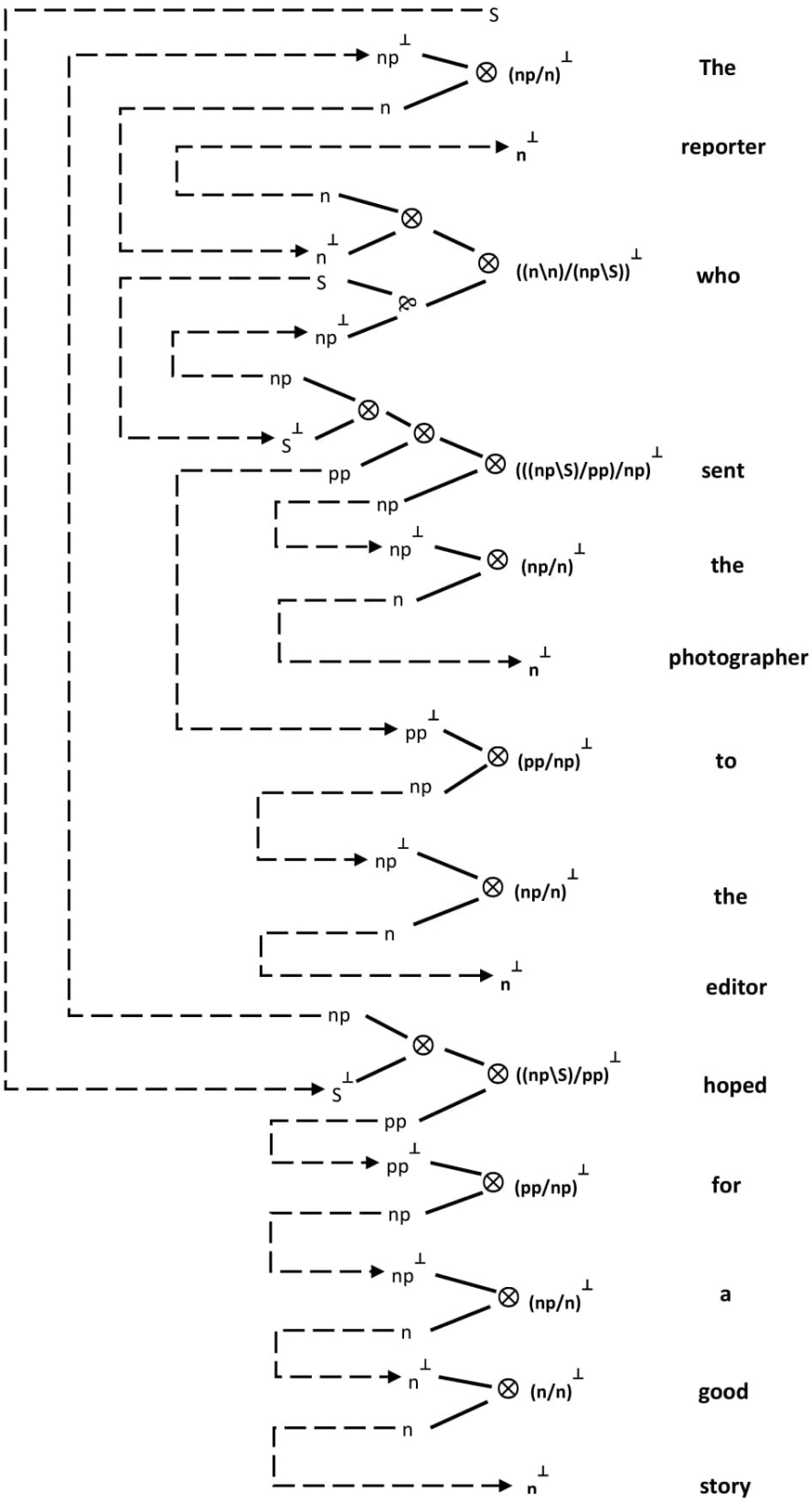


FIGURE 5.2: Proof net analyses for 5.3a (subject-extracted relative clause).

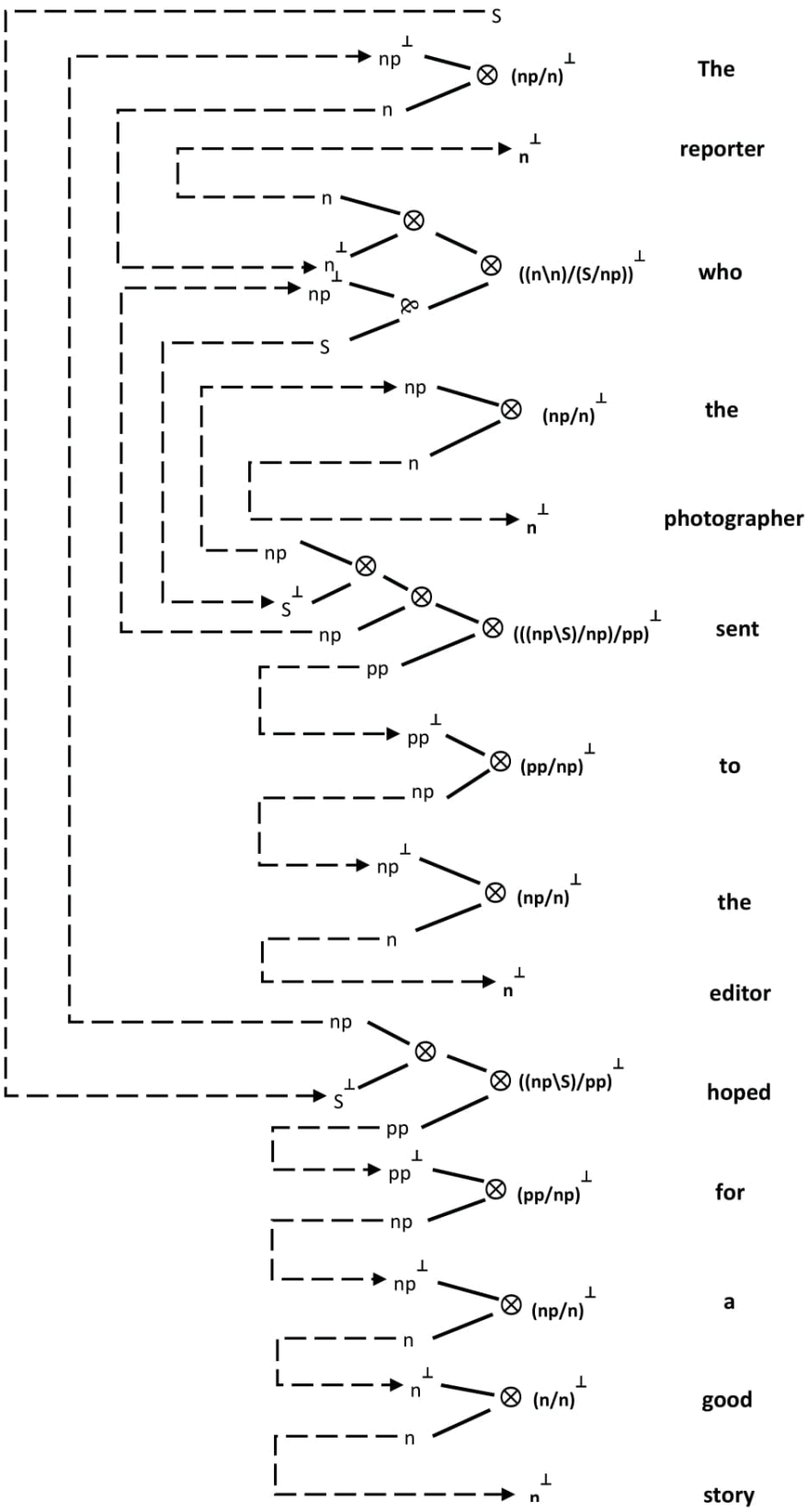


FIGURE 5.3: Proof net analyses for 5.3b (object-extracted relative clause).

5.4 A New Proposal: Distance Locality-Based Complexity Profiling

As we discussed, IDT-based complexity profiling is a distance-based measurement. However, it is not a referent-sensitive criterion and due to this fact, it cannot support some of the linguistic phenomena such as structures with embedded pronouns. One plausible strategy to overcome this lack is introducing DLT-based complexity profiling. This will allow us to have a referent-sensitive measurement. In this section, we provide the precise definitions of our DLT-based proposal on the basis of the categorial proof nets. Here they are:

Definition 5.4. A word w is said to be a discourse referent whenever it is a *proper noun*, *common noun* or *verb*.

Definition 5.5. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let $c - c'$ be an axiom in π such that $c \in C_i$ and $c' \in C_j$ $(i, j \in [1, n])$. We define the **length** of axiom $c - c'$ as the integer $i + 1 - j$.

Definition 5.6. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i, and let consider axioms $c - c'$ with c in C_{i_0} and c' in some C_{i_0-k}. Let us consider the largest k for which such an axiom exists — this is the longest axiom starting from C_{i_0} with the previous definition. The dependency locality number of C_{i_0} in π, written as $DL_\pi(C_{i_0})$, is the number of discourse referent words between $w_{i_0} : C_{i_0}$ and $w_{i_0-k} : C_{i_0-k}$. The bound words, i.e. $w_{i_0} : C_{i_0}$ and $w_{i_0-k} : C_{i_0-k}$ should also be counted. Alternatively, it may be viewed as $k + 1$ minus the number of non-discourse references among those $k + 1$ words.

Definition 5.7. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. We define the DLT-based linguistic complexity of π, written $f_{dlt}(\pi)$ by $f_{dlt}(\pi) = (1 + \sum_{i=1}^{n} DL_\pi(c_i))^{-1}$.

Definition 5.8. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is DLT-preferred to π_j whenever $f_{dlt}(\pi_i) > f_{dlt}(\pi_j)$.

Example 5.4. We apply our new metric on examples 5.1b and 5.1c; repeated here as 5.4a and 5.4b.

5.4a. The reporter [who the senator [who John met] attacked] disliked the editor].

5.4b. The reporter [who the senator [who I met] attacked] disliked the editor].

Figure 5.5 and 5.7 shows the relevant proof net for 5.4a and 5.4b, respectively. The proof nets for both examples are the same except a difference in one of the lexicons in each example, i.e. *John* and *I*. Figure 5.6 shows the accumulative chart-based

representation of our measurement for each example. The axis Y shows the accumulative sum of dependency locality function applied to each category in axis X. The quick analysis of the profiles shows the total complexity numbers 14 and 11 for 5.4a and 5.4b, respectively. This correctly predicts the preference of example 5.4b over 5.4a which was not possible in the IDT-based approaches. The measurement for dependency locality number is quite straightforward. As an example, we calculate the dependency locality number for the word *attacked* in figure 5.7 for 5.4b. We can find the longest axiom link starting from *attacked* and ended to its uppermost category, namely, *who*. Then, we count the number of discourse referents intervened in the axiom link, which is actually three; namely, *attacked*, *met* and *senator*.

5.5 Evaluation of the New Proposal against other Linguistic Phenomena

We can evaluate our proposal for measuring the linguistic complexity against other linguistic phenomena. As we will see, the new metric supports both referent-sensitive and some of the non-referent-sensitive phenomena[5]. We start with subject/object extracted relative clause in the examples 5.3a and 5.3b. The figures 5.2 and 5.3 illustrate the proof net analyses for 5.3a and 5.3b, respectively. We can now compare the complexities of 5.3a and 5.3b. The accumulative complexity chart is illustrated in Figure 5.4. The analysis of the profiles shows the total complexity numbers 7 and 8 for 5.3a and 5.3b, respectively. The interpretation of these numbers is that subject-extracted relative clause is less complex compared to the object-extracted relative clause. These measurements account for reading-time effects as performed with the actual reading times for participants in a self-paced reading experiment [GK98].

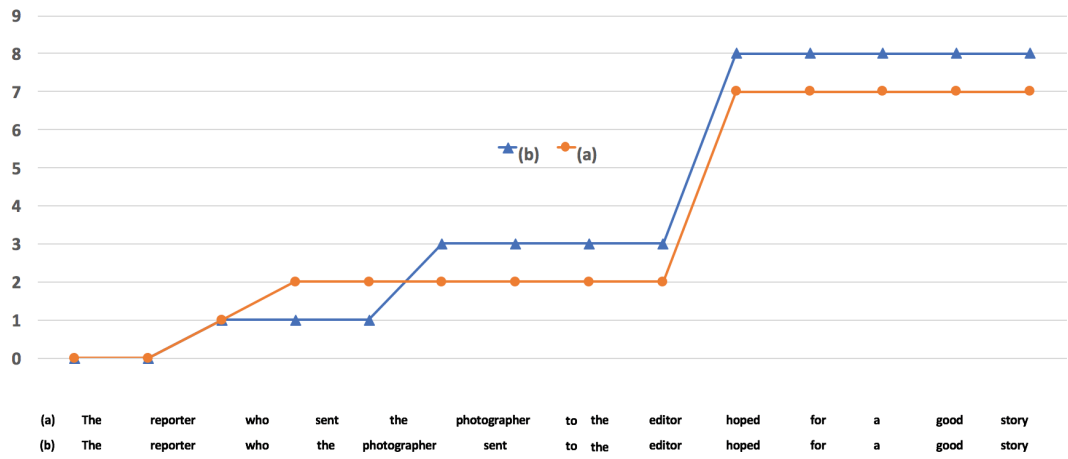


FIGURE 5.4: Accumulative DLT-based Complexity Profiles for 5.3a and 5.3b.

The Garden pathing [Bev70] is illustrated by the following examples:

Example 5.5.

5.5a. The horse raced past the barn.

[5]We have not covered the problem of the ranking valid semantic meanings of a given multiple-quantifier sentence in this study. [For more details see section 5.6.]

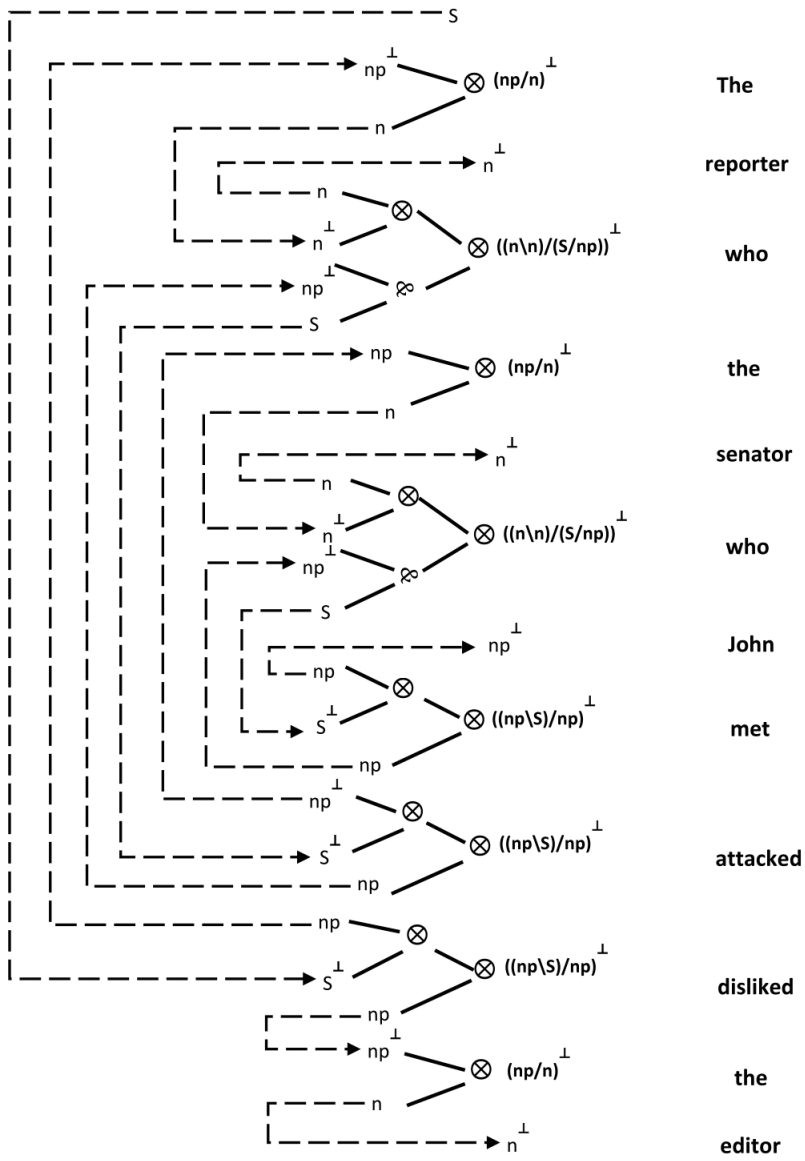


FIGURE 5.5: Proof net analysis for the example 5.4a.

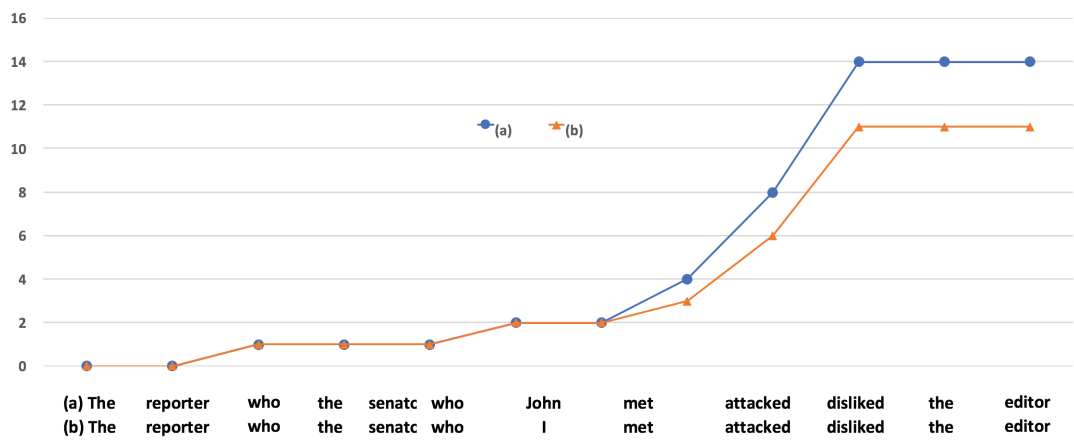


FIGURE 5.6: Accumulative DLT-based Complexity Profiles for 5.4a and 5.4b.

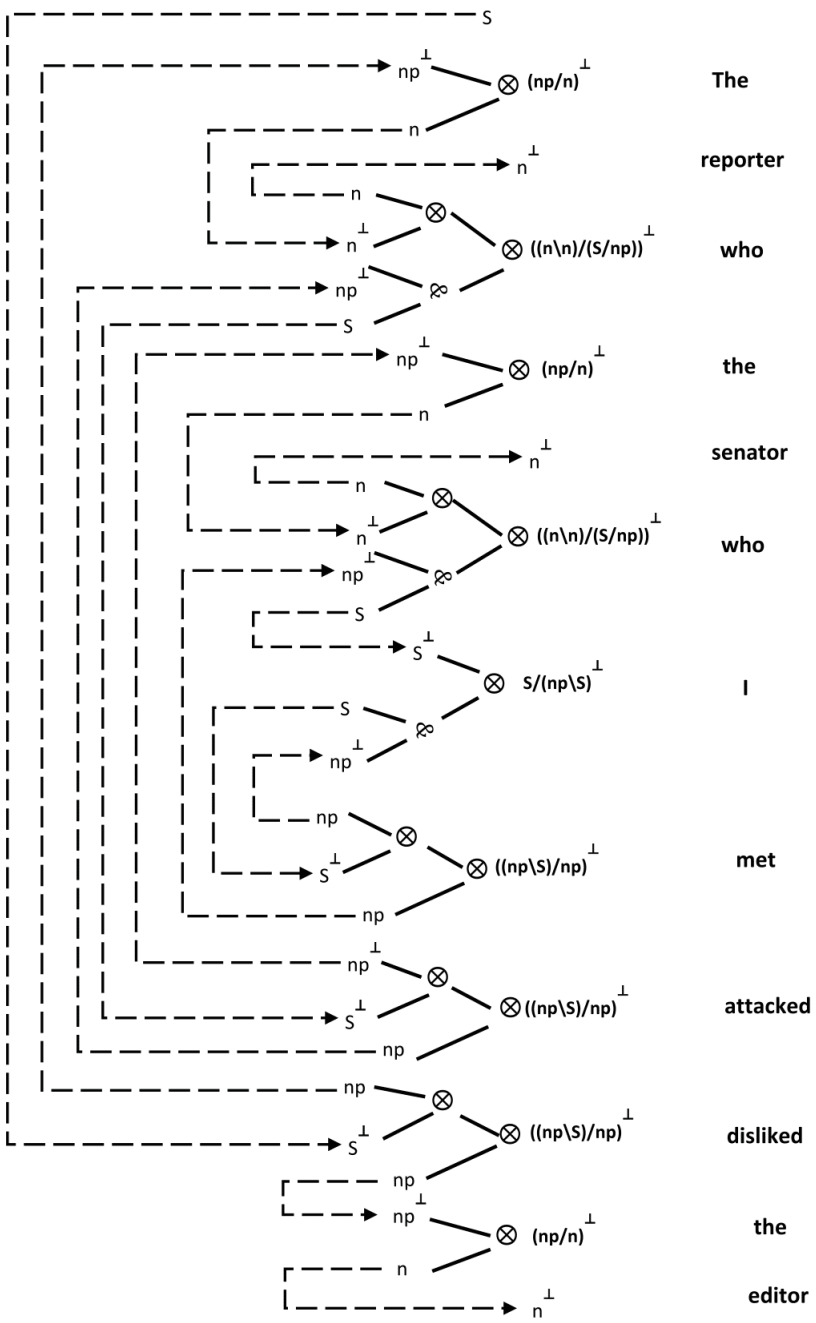


FIGURE 5.7: Proof net analysis for the example 5.4b.

5.5b. ?The horse raced past the barn fell.

Although 5.5b is grammatical it is perceived ungrammatical due to strong tendency to interpret the initial segments as in 5.5a. The sentence 5.5b leads the reader toward a seemingly familiar meaning, namely 5.5a, that is actually not the one intended. This often requires re-reading so that the meaning may be fully grasped after careful parsing. The difficulty in correctly parsing the sentence 5.5b results from the fact that *race* can be interpreted transitively or intransitively. The figure 5.10 illustrates the proof net analyses for 5.5a and 5.5b. Figure 5.8 shows the complexity profiles in which the total complexity numbers are 2 and 7 for 5.5a and 5.5b, respectively. This correctly predicts the high level of complexity in 5.5b and the preferred reading of 5.5a comparing to 5.5b as it happens in the real sentence comprehension.

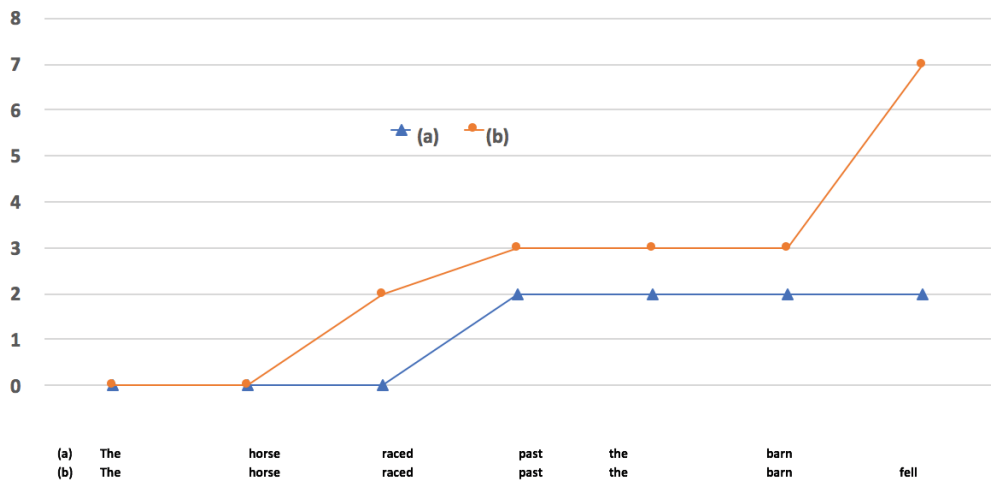


FIGURE 5.8: Accumulative DLT-based Complexity Profiles for 5.5a and 5.5b

The unacceptability of center embedding phenomena is illustrated by the examples 5.6a and 5.6b. The example 5.6a shows object relativization which exhibits deterioration in acceptability while the sentence 5.6b exhibits little variation in acceptability since it carries subject relativization [Cho14, Chap. 1]. This linguistic phenomenon can be captured in our model. The figures 5.11 and 5.12 illustrates the proof net analyses for 5.6a and 5.6b, respectively. Figure 5.9 shows the complexity profiles in which the total complexity numbers are 14 and 10 for 5.6a and 5.6b, respectively. Our proposal correctly predicts the complexity of 5.6a over 5.6b.

Example 5.6.

5.6a. ?The cheese that the rat that the cat saw ate stank.

5.6b. The dog that chased the cat that saw the rat barked.

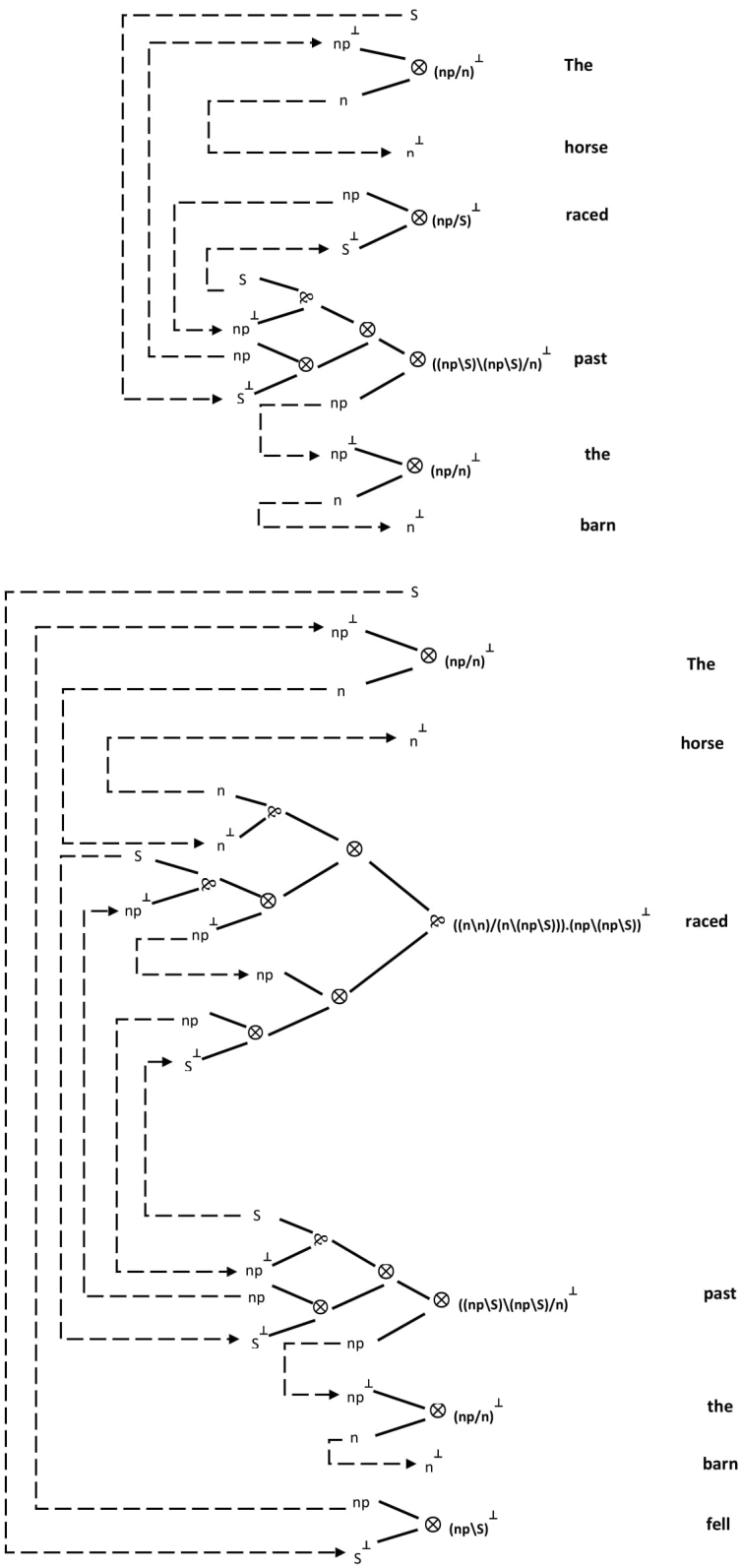


FIGURE 5.10: Proof net analyses for 5.5a located in top (first attempt reading) and 5.5b in bottom (full garden path sentence).

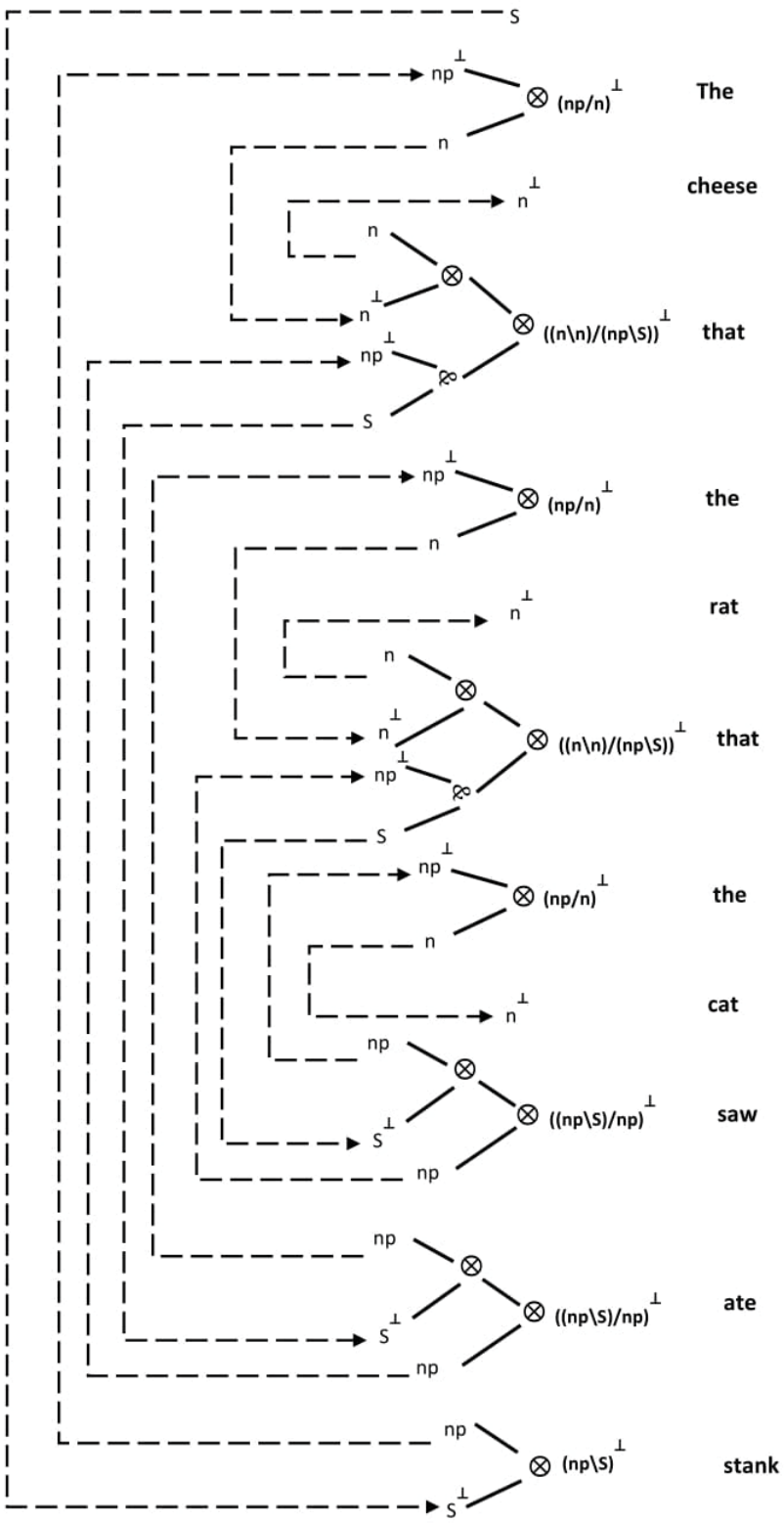


FIGURE 5.11: Proof net analyses for 5.6a (object relativization).

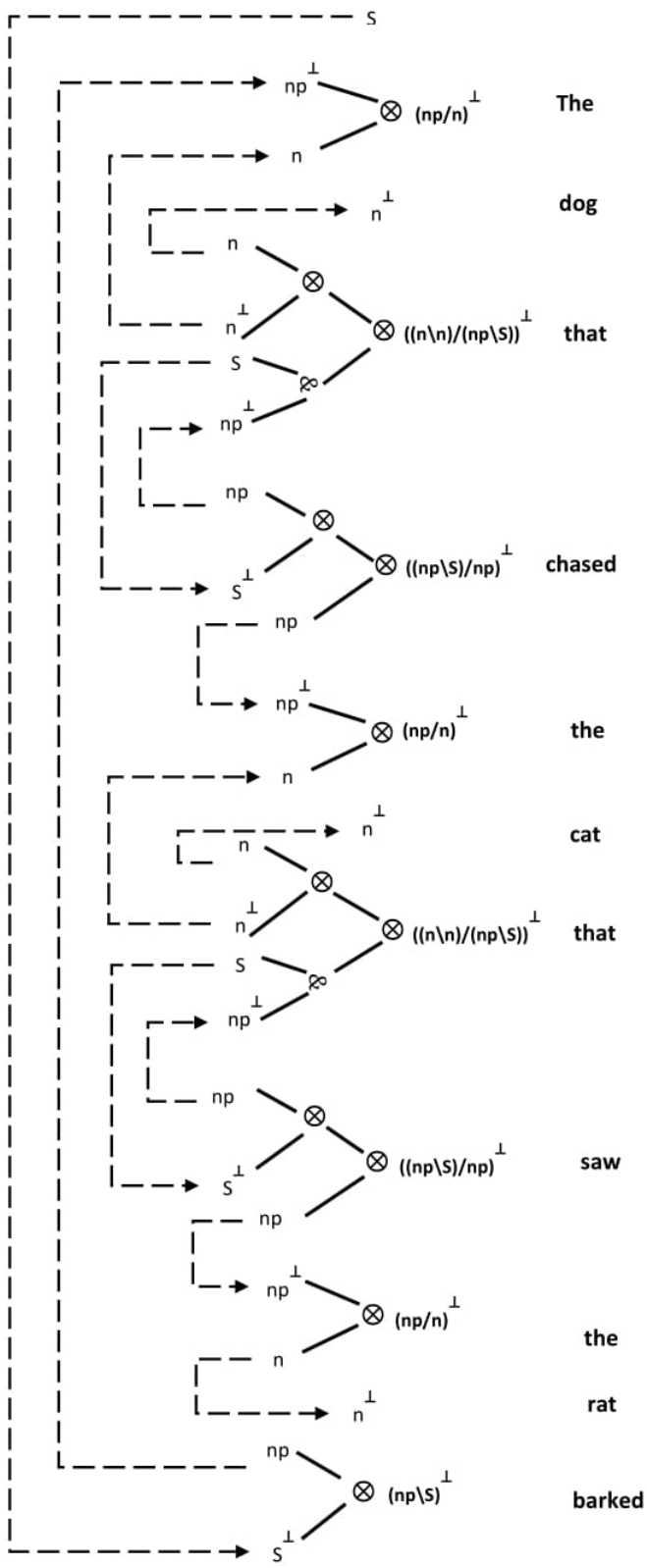


FIGURE 5.12: Proof net analyses for 5.6b (subject relativization).

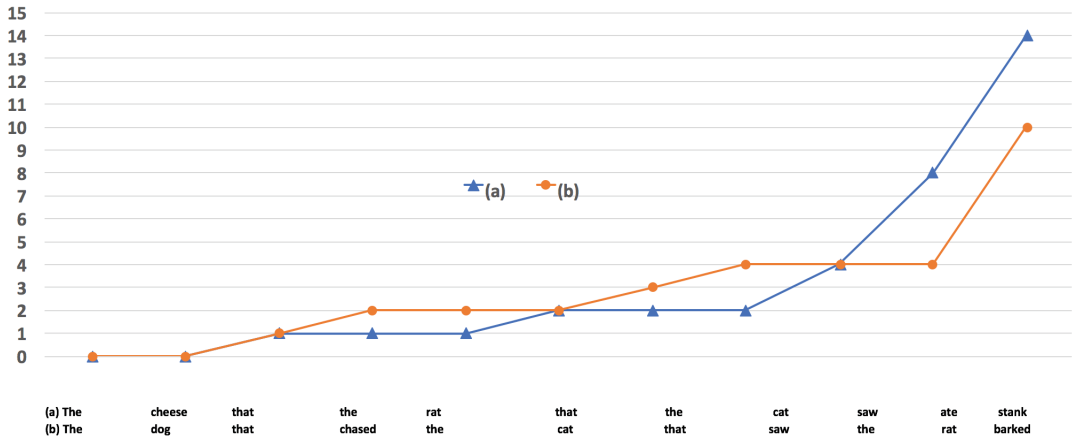


FIGURE 5.12: Accumulative DLT-based Complexity Profiles for 5.6a and 5.6b

Examples 5.7 and 5.8 carry different readings due to the syntactical ambiguity that exists in each of them. As for the example 5.7, the preference is for the lowest attachment [Kim73]. The three profiles in figures 5.13, 5.14 and 5.15 show the highest, middle and lowest attachment of the adverb *today* to the verbs *said*, *believes* and *fell*, respectively. As it is illustrated in the figure 5.9, our model can predict the preference of the 5.7a over 5.7b, and 5.7b over 5.7c. In other words, the lower attachment of the adverb, the higher the preference [Kim73].

Example 5.7. Joe said that Marthus believes that Ingrid fell today.

Example 5.8. The book that shocked Mary's title.

As for the example 5.8, two proof nets in the figure 5.17 show the lowest and highest attachment of the phrase *'s title* to the sentence *the book that shocked Mary*. As it is illustrated in the figure 5.18, our model can predict the preference of the right interpretation in the figure 5.17 over the left one, despite the fact that the right one is non-sensical [Mor00].

The last linguistic performance phenomena that we want to discuss is Passive Paraphrases Acceptability [Mor00] illustrated by examples 5.9a and 5.9b. Notice that the DLT-based complexity profile of the 5.9a is lower even though the number of the sentences and the axiom links are more comparing to 5.9b. The real preference is on the syntactic forms in which 5.9a is preferred to 5.9b. The relevant proof nets and the accumulated complexity profiles are illustrated in the figures 5.19-5.20 and 5.21, respectively.

Example 5.9.

5.9a. Ingrid was astonished that Jack was surprised that two plus two equals four.

5.9b. ?That that two plus two equals four surprised Jack astonished Ingrid.

5.6 Limitations

As we saw, our DLT-based complexity profiling is quite successful in predicting some linguistic performance phenomena. We also show that the referent-sensitive phenomenon cannot be treated by IDT-based approach while our approach could do so. Nevertheless, there are two limitations in our approach as well. The first one

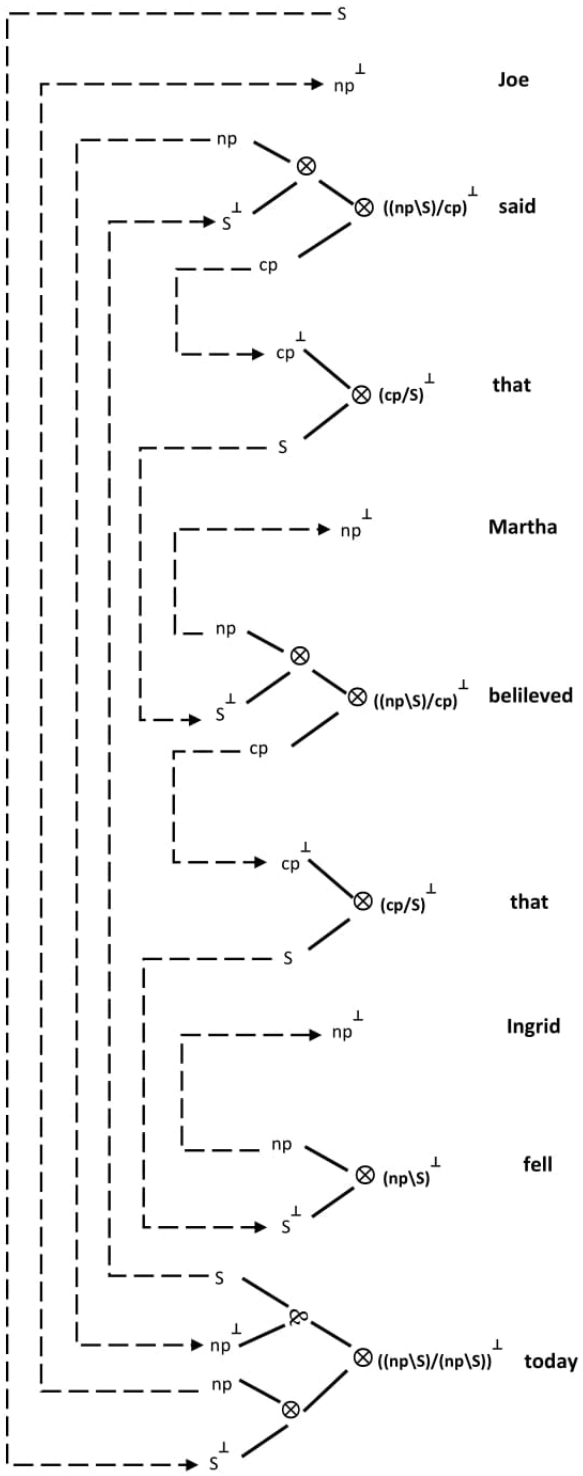


FIGURE 5.13: Proof net analyses for 5.7 with highest adverbal attachments.

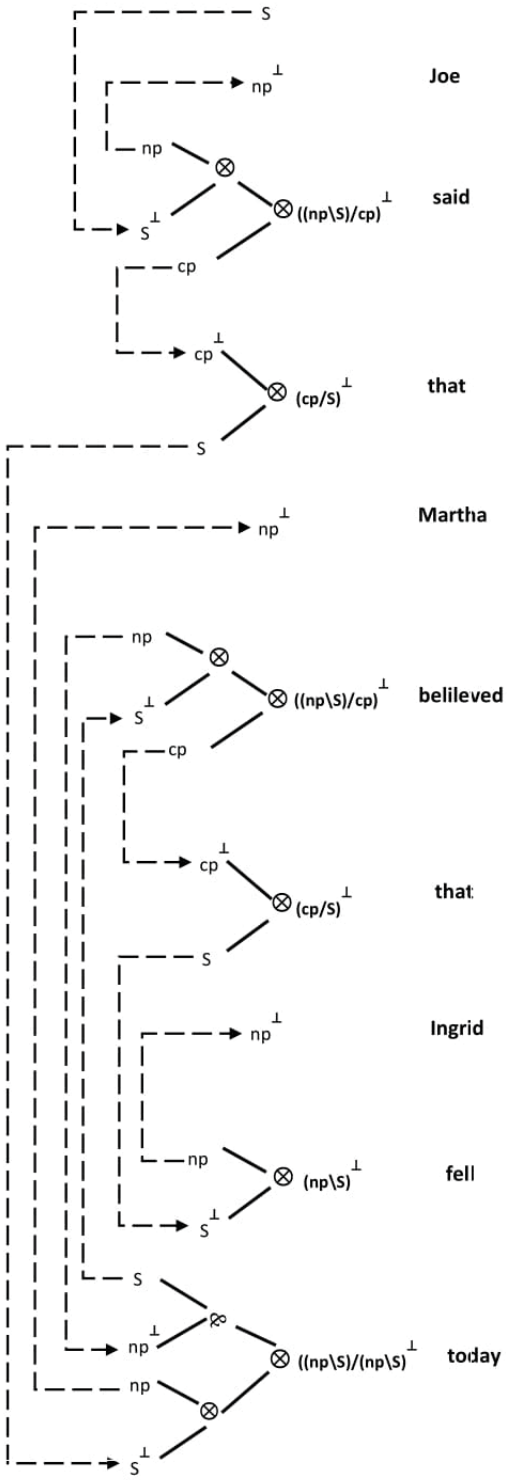


FIGURE 5.14: Proof net analyses for 5.7 with middle adverbal attachments.

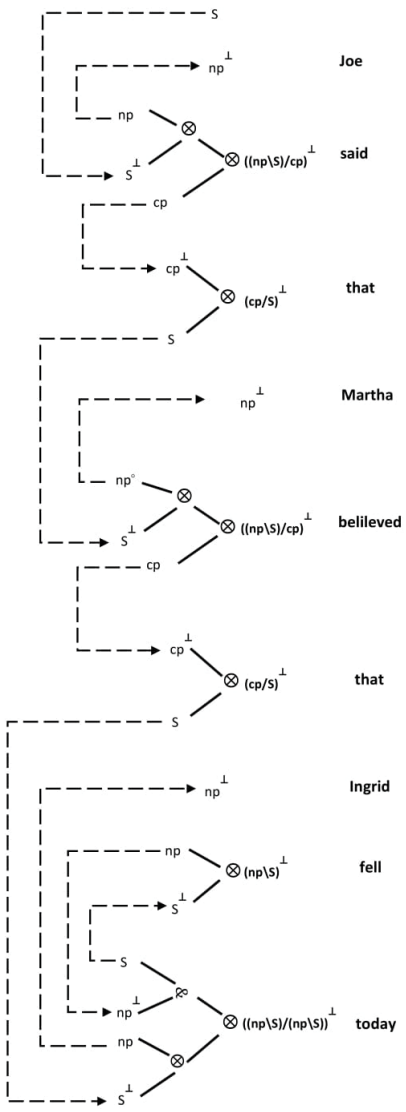


FIGURE 5.15: Proof net analyses for 5.7 with lowest adverbal attachments.

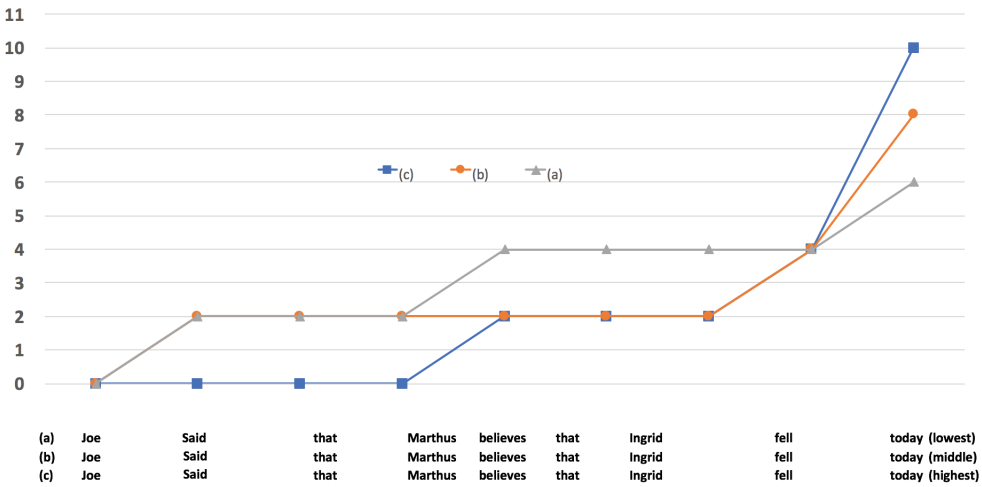


FIGURE 5.16: Accumulative DLT-based complexity profiles for three readings of 5.7

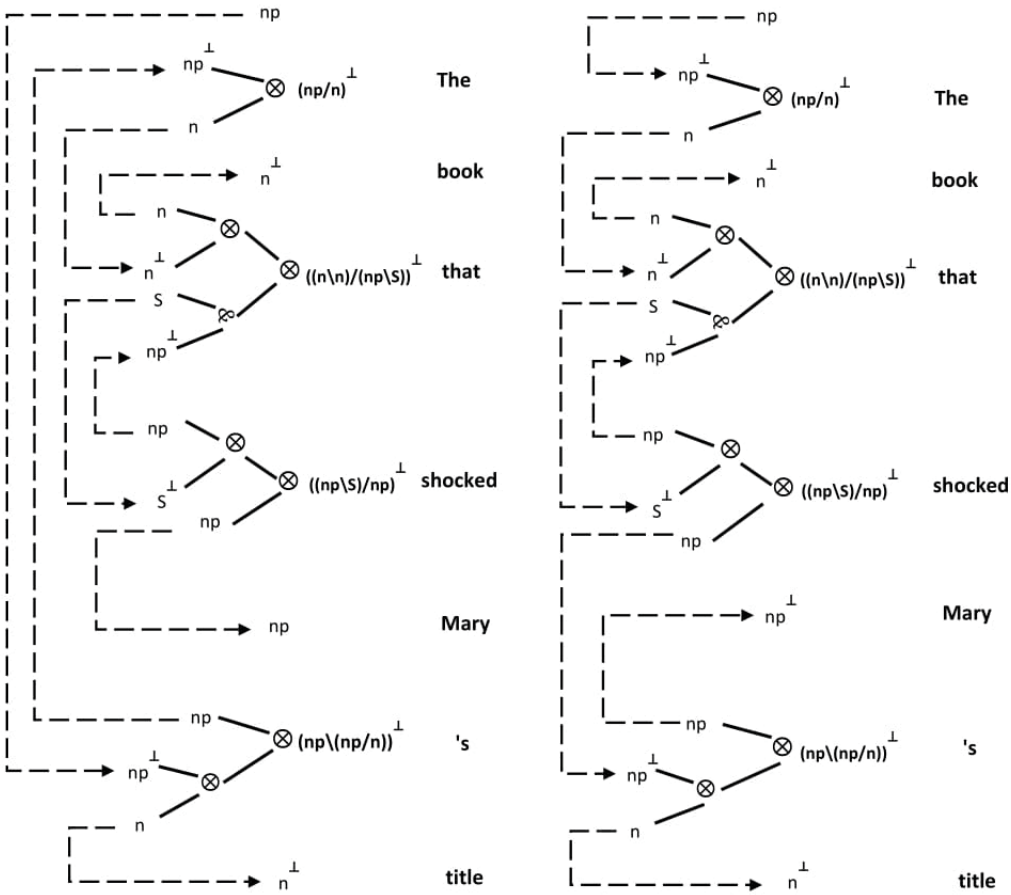


FIGURE 5.17: Proof net analyses for 5.8 with sensical (left) and non-sensical (right) interpretations.

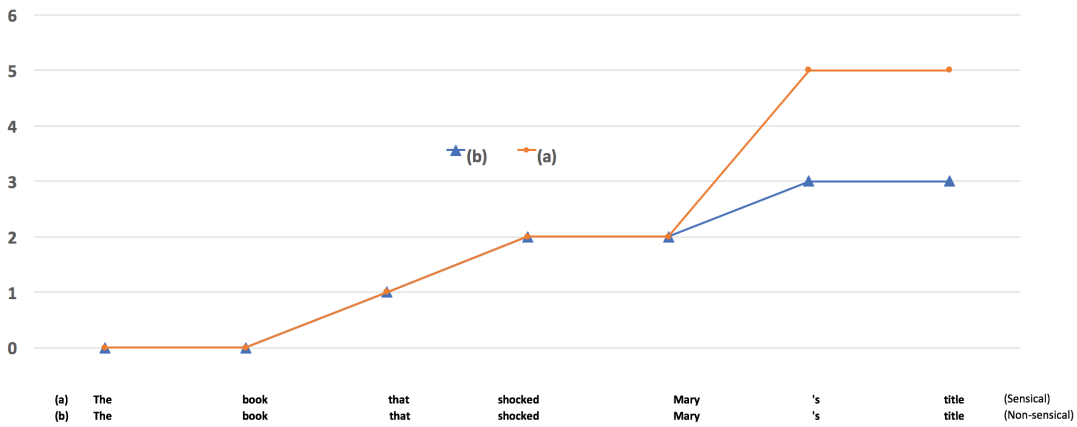


FIGURE 5.18: Accumulative DLT-based complexity profiles for two readings of 5.8

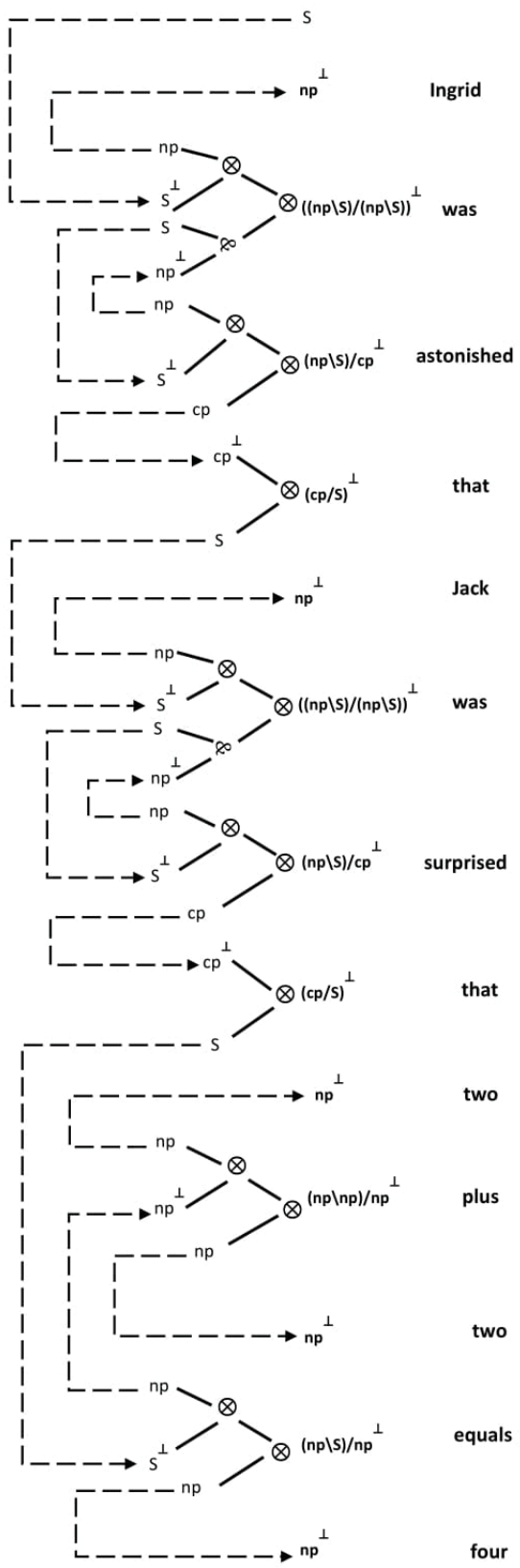


FIGURE 5.19: Proof net analyses for 5.9a .

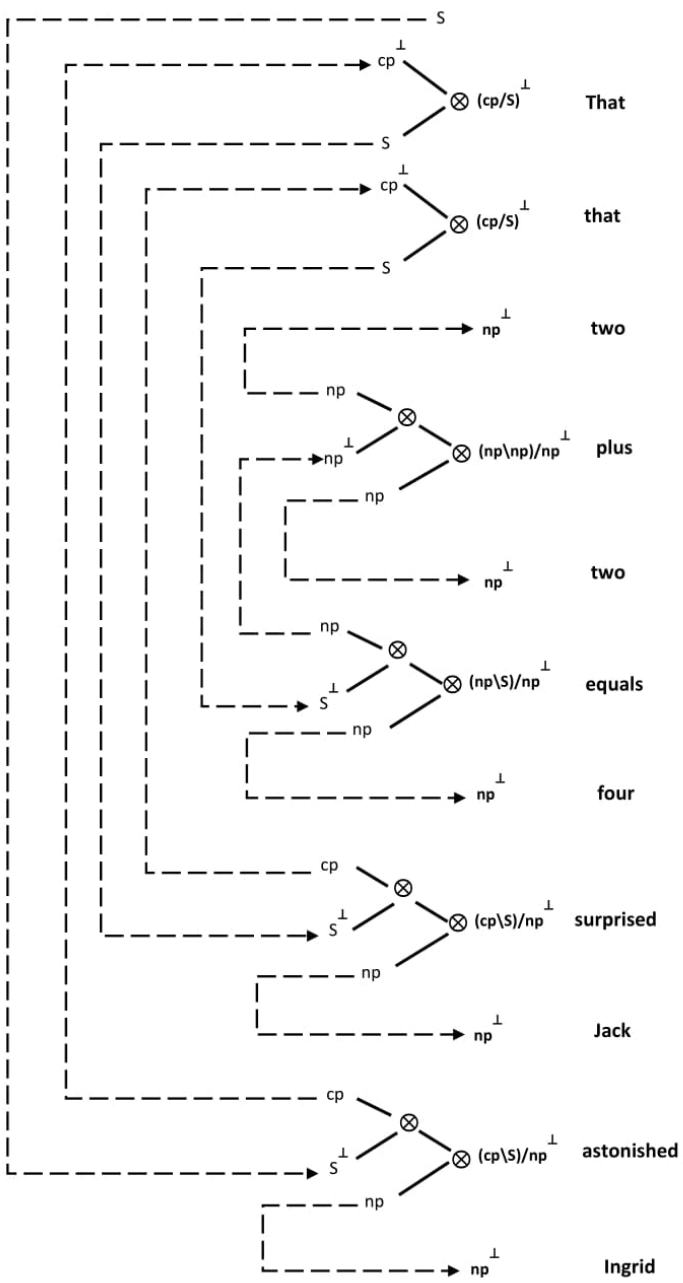


FIGURE 5.20: Proof net analyses for 5.9b .

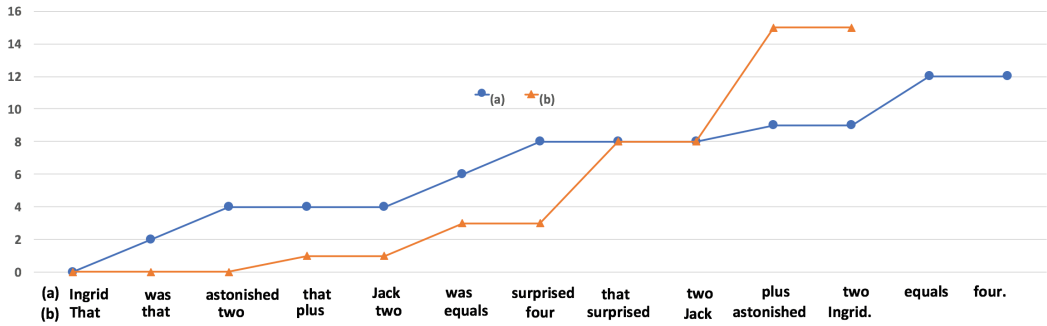


FIGURE 5.21: Accumulative DLT-based complexity profiles for 5.9a and 5.9b

is the problem of ranking valid semantic meanings of a given multiple-quantifier sentence which cannot be supported by our proposal. We studied in the chapter 3 the same problem in the IDT-based approach when dealing with some type of the expressions such as sentence-modifier adverbials and nested sentences. Our experiment shows that the same limitations hold in our DLT-based approach. Thus, both IDT-based and DLT-based complexity profiling cannot correctly predict ranking the quantifier scoping problem. Hopefully, this can be treated with the hybrid models (discussed in the chapter 3) in which Hilbert's epsilon and tau [Hil22; CPR17] are exploited in order to neutralize the quantifier effect, and after that, the complexity is measured based on the penalty of the quantifiers re-ordering. This treatment for DLT-based approach will be briefly introduced in the chapter 7.

The second drawback of the DLT-based motivated approaches is that they are not applicable cross-linguistically for human parsing processes. One study [Vas+05] has shown the failure of DLT-based approaches in modeling some human performances in the Hindi language. DLT uses the distance hypothesis: the increasing distance between dependents and heads results in increased processing difficulty. This hypothesis is not a universal linguistic feature as it is claimed [Vas+05]. This cross-linguistically problem in DLT-based approaches, has motivated the activation-based models [LV05] which are formalized in computational form in the Adaptive Control of Thought-Rational (ACT-R) architecture [And+04]. This theory construes parsing as an efficient series of the guided memory retrievals and not parsing as proof net construction in a way we have practiced in this chapter. Nevertheless, activation-based approaches cannot properly treat the structures with embedded pronouns [LV05]. Moreover, it has no systematical approach for deriving automatically meaning representations in logical formulas as one is generally capable to do so in the Categorical Grammar frameworks and specifically in our proposed DLT-based complexity profiling.

5.7 Conclusion and Possible Extensions

In this chapter, we explored how our DLT-based complexity profiling on proof nets can successfully support a number of linguistic processing phenomena such as structures with embedded pronouns, garden paths, unacceptability of center embedding, preference for lower attachment, and passive paraphrases acceptability. Our proposal gets closer to the modern psycholinguistic theories while it adds the good features of Categorial Grammars such as the deep semantic analysis. We have also shown that IDT-based method could not support referent-sensitive linguistic performance phenomena. This was one of the main reasons for introducing the DLT-based complexity profiling technique within the framework of Lambek calculus.

It is worth mentioning that Gibson's DLT-based complexity can be applied to other grammatical formalisms such as Property Grammar [Bla16], Pregroup Grammar [Lam97] and Categorial Dependency Grammar (CDG) [Dik04]. To the best of our knowledge, Property Grammar is the only case in which both DLT-based and IDT-based approaches [Bla11b; Bla11a] are practiced, although they are not evaluated for all the available linguistic performance phenomena. Property Grammar has the advantage of its robustness when dealing with non-canonical sentences [PS01]. Meanwhile, it has no straightforward syntax-semantic interface. As for Pregroup Grammar, there is a study [Sad08] for the IDT-based approach for linguistic complexity and no DLT-based application yet. It is technically possible to apply

DLT-based measurement to Pregroup Grammar, nevertheless, we preferred to work with categorial proof nets, since it has a straightforward syntactic-semantic interface, learnability property from positive sentences [BP90; Kan98] and wide coverage syntax/semantic parsing [Moo10], which Pregroup Grammar lacks. CDG discriminates very clearly between local and distant word driven dependencies. Local dependencies are defined in terms of classical categorial grammar terms, while the distant dependencies are defined in term of polarized valencies. Technically, it is quite possible to have both the IDT-based and DLT-based complexity measurements on a fragment of CDG that deals with local dependency. However, the main interest of CDG— which is incorporating the non-projective fragment of natural language— cannot be applied in the distance-based proposal. This is a serious lack of the application of proof nets in CDG. Nevertheless, we do not consider all the mentioned problems in Property Grammar, Pregroup Grammar and CDG as something intractable, we consider them as open questions that can be explored. This obviously goes far beyond our current research and demands new research.

There are three straightforward extensions for our study and research, that are already investigated in this dissertation:

- We can bridge our model with existing study [CM17] to overcome the problem of ranking quantifier scoping, which our proposal already has. As we discussed, we can exploit Hilbert's epsilon and tau operators [Hil22; CPR17] for neutralizing the quantifier effect and making possible the complexity measurement by the penalty cost of the quantifiers re-ordering. We will see the details of this method in the chapter 6.

- Another important direction is to take into account not only the axioms of the proof-nets but also the logical structure, i.e., par-links, tensor-links and the correctness criterion. This is important indeed because this structure is needed to compute the logical form (semantics) from the syntactic structure given by proof nets. For instance, nesting Lambek slashes (that are linear implications, and therefore par-links in the proof net) corresponds to higher order semantic constructions (e.g. predicates of predicates) and consequently this nesting of par-links increases the complexity of the syntactic and semantic human processing.

- It is possible to combine our method with the researches in other directions. One potential candidate is the task of sentence correction/completion in Lambek Calculus [Mir] and also in natural language processing in general. Given an incomplete utterance, we might have different potential proof nets either over-generated by a framework or properly generated. The sub-task of ranking these candidates is aligned with our research. We will investigate this approach in the chapters 6 and 7.

There are more general ideas, not so far from our current study, that can be considered for our future research. One possible direction is to take into account other possible factors in linguistic preferences such as common sense and lexical knowledge. This is important since these factors override the preferences in some cases and affect the complexity. For this purpose, we can exploit Montagovian Generative Lexicon (MGL) framework [Ret14] that uses multi-sorted logical formulas. This choice would let us apply semantic distance approaches [Nag94] or exploit weighted relations [Laf+p2] provided in lexical semantic networks such as JeuxDeMots (JDM) [Laf07; Cha+15]. With these approaches, we can create and enrich wide-coverage,

weighted, lexical resources such as the crowd-sourced JDM in order to infer linguistic features. Another worthwhile direction of study would be to create a dataset annotated for human-preferred readings of various natural utterances. Consequently, we could have more experimental data in order to evaluate/extend our new proposals over the test-suite. This experiment would let us integrate other aspects such as lexical semantics to our model easily with the online evaluation.

Chapter 6

Modeling Meanings Preferences IV: Ranking Incomplete Sentence Interpretations

6.1 Introduction [1]

As we discussed, an utterance may yield several readings either from ambiguities (lexical, syntactical and semantical) or from syntactic incompleteness. The incomplete utterances can be linguistically considered as a type of non-canonical utterances [PS01] and they can be caused by different reasons such as missing categories, extra categories, swap categories, misused categories, or any possible combination of mentioned reasons. The focus in this chapter is only on incomplete utterances with missing categories. To make our research problem clear we should emphasize that by the missing category one can generally suppose two things: a missed word in an incomplete sentence or an unknown word in a sentence with no proper category assignment. In this section, we especially deal with the problem of missing categories for some missing words in an incomplete sentence. Moreover, we assume that the position of the word in the sentence is unknown.[2]

Analysis of incomplete utterances has been the subject of study in rule-based, data-driven and statistical approaches in computational linguistics [Lea+10]. The scope of automatic error detection/fixation and robust parsing is so wide and different mechanisms are introduced by researchers such as[3]: over-generating and ranking parse trees [DM93], imposing ranking constraints on grammatical rule [Mel89], introducing "mal-rules" [SM98], relaxing constraints in feature unification [VC95], modeling grammar as model-theoretic constraints [Bla00a; DB04a; Pro10], and finally, sub-tree parse fitting for non-parsable utterances. [Jen+83]. All the mentioned approaches have their own advantages and address the same issue that we have defined. However, the formalisms that are introduced in them is not the same as what is used in this section.

[1] The material in the section 6.2 of this chapter is derived in large part from [Mir] which is the author's work. The rest is the author's work which is not published yet.

[2] We should re-emphasize that we model only a special class of the non-canonical sentences which are sentences with missing categories. This means that the order of the words in the sentence is adequate, but, some words are missing and we need them to complete the meaning of the sentence. Thus, we will not take into account other problems such as extra categories or misused categories or even other aspects of non-canonical sentences which cover the language use or pragmatic.

[3] See [Lea+10, Chapter 1] for detailed descriptions.

Basic Categorial Grammars or AB grammars is a formalism which is used in this research. It is a family of the categorial grammars. This choice is based on the requirement to bridge the incomplete utterances with semantical readings. There are a number of reasons to explain why categorial grammars are suitable for our specific research topic and this was discussed in the previous chapters, to sum up: Firstly, categorial grammars have a very nice correspondence with semantics and they are very suitable for syntactic-semantic interface modeling; they can represent semantic ambiguities such as quantifier scope and negation [MR12b, Chapter 3] which does not exist in shallow approaches; secondly, they enjoy learnability property from positive sentences which converge after learning a fixed number of sentences by using unification techniques [BP90; Kan98]; thirdly, by applying a left-to-right incremental procedure and proof net construction for categorial grammar we can correctly predict a wide variety of human performance phenomena as we discussed in the chapter 5; finally, there is a wide-coverage syntactic and semantic parsing for French language, named Grail, that allows researchers to design and experiment with multimodal categorial grammars which include AB grammars as well. [Moo10]. Grail is a modern, flexible and robust parser/automated theorem prover that has around 900 different category types for French language and works efficiently with the newspaper articles. Categorial grammar is extracted semi-automatically from the Paris 7 tree-bank and its semantic lexicon maps combinations of words, part-of-speech tags, and formulas to Discourse Representation Structures [KR93].[4]

The main task we are pursuing is to find the potential candidates for completing the sentence using categorial grammars. Also, we would like to introduce different psycholinguistic measurements for categorial proof nets in order to quantify the human performance phenomena. We will show why modeling these ranking is not as straightforward as it may be seen.

In the first place, we introduce two algorithms for resolving incomplete utterances: (i) the first algorithm is based on using AB grammars as the basis of our syntactic formalism on the one hand, and adopting the notion of unification that exists in RG algorithm (learning Rigid AB grammars) [BP90; Bus87] plus some dynamic programming technique on the other hand. This approach, to the best of our knowledge, is a new research technique that can efficiently deal with incomplete utterances with missing categories while benefiting from the good properties in categorial grammars. As we will see, the time complexity for the algorithm that we use to find one missing category with n number of words is $O(n^4)$. (ii) the second algorithm employs Constraint Handling Rules (=CHR) which ends up with some potential candidate(s) for correcting the utterances. In the second place, we introduce new measurement on categorial proof nets such as *Satisfaction Ratio* and *Activation Theory*, and we will also review Gibson's distance-based approaches which we introduced in the chapters 3 and 5.

The rest of the chapter is organized as follows: section 6.2 explains an algorithm with $O(n^4)$ time complexity for finding a missing category in an incomplete utterance by using unification technique as when learning categorial grammars, and dynamic programming as in Cocke–Younger–Kasami algorithm. In section 6.3, we explain another algorithm which can find more than one missing categories in an

[4] As for the behaviour of the Grail parser on the sentences containing an unknown word we should say that the wide-coverage version of the Grail parser uses a probabilistic model to propose one or more of the contextually most likely options. It is worth mentioning that our proposal can be used for finding missed words; although it can be adopted for finding categories for unknown words as well.

incomplete sentence. Though, we have this at the cost of having an exponential time complexity. We will see some of the good properties of this algorithm. In section 6.4, we define new metrics for the possible candidates of a given incomplete sentence which are motivated from *Satisfaction Ratio* and *Activation Theory*. In the last section, we conclude and we explain possible future works.

6.2 Sentence Completion: Algorithm A

This section introduces an efficient algorithm with $O(n^4)$ time complexity for finding a missing category in an incomplete utterance by using unification technique as when learning categorial grammars, and dynamic programming as in Cocke–Younger–Kasami algorithm. Using syntax/semantic interface of categorial grammar, this work can be used for deriving possible semantic readings of an incomplete utterance. We will illustrate the problem with some running examples.

6.2.1 Definitions

In this subsection, some essential notions are discussed very quickly. We need these notions and definitions for understanding the underlying technique that is used in our algorithm.

Definition 6.1. Categories:

Taking Lambek's notation [Lam58], categories in AB grammars can be written as follows [MR12b, Chapter 1]:

$$L ::= P \mid (L\backslash L) \mid (L/L)$$

where P is the set of primitive category (such as np and S) while $B\backslash A$ and A/B can be interpreted as the functors that take B as their arguments on its left and right-hand side respectively; in the all mentioned cases, the result would be A. A function Lex is a lexicon which maps words to a finite set of categories. An expression, that is a sequence of words or terminals $w_1, \cdots, w_n$ is of category u whenever there exists for each w_i a category u_i in $Lex(w_i)$ such that $u_1 \cdots u_n \rightarrow u$ with the left or right elimination rule patterns, namely $(A/B)B \rightarrow A$ and $B(B\backslash A) \rightarrow A$.

It is worth mentioning that variable categories are also used such as $x_1, x_2, \cdots, x_n$ (or $y_1, y_2, \cdots, y_n$) in addition to above fixed category–or what can be properly named fixed learned categories with syntactic labels– for our future purposes in unification phase. See following example a lexicon and the related process which ends up to S:

Example 6.1. Lexicon:

Jean : np	aime: $(np\backslash S)/np$	Marie: np

Jean aime Marie $\Longrightarrow np,\ (np\backslash S)/np,\ np \Longrightarrow np,\ (np\backslash S) \Longrightarrow S$

Definition 6.2. Structural trees with variable categories:

It is basically a tree-structured proof representation with variable categories instead of having fixed categories. The only exception is fixed category S in the root

of the tree which is naturally expected. Arguments will have unique variable while the functors would get the argument category as its sub-formula in the right or left side. All the possible structural trees of a sequence with 3 words are represented in the figure 6.1. For convenient, when we use the term structural tree we mean the structural tree with variable categories.

It can be easily noticed that the number of possible trees for a sequence of string with n length would be $catalan(n-1) * 2^{n-1}$ in which $catalan(n-1)$ is the number of possible binary trees and 2^{n-1} is the number of possible operations for \ and / over each binary sub-trees.[5] In the case of a sentence with 3 unspecific words, the number of all possible structural trees is 8 (= $catalan(3-1) * 2^2$) as it can be observed in the figure 6.1.

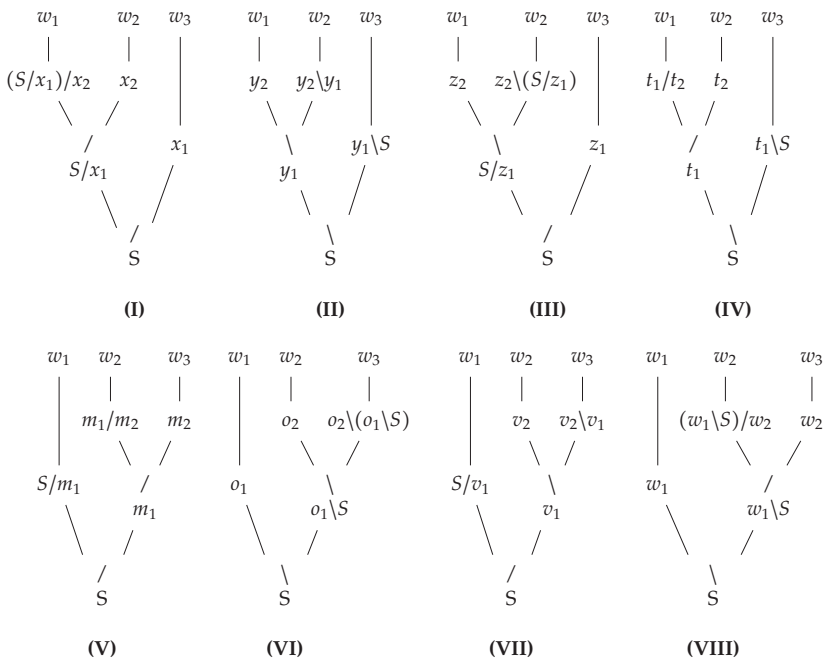


FIGURE 6.1: Possible structural trees of a sequence $w_1 w_2 w_3$

Words	Fixed learned categories	I	II	III	IV	V	VI	VII	VIII
Jean	np	$(S/x_1)/x_2$	y_2	z_2	t_1/t_2	S/m_1	o_1	S/v_1	w_1
aime	$(np\backslash S)/np$	x_2	$y_2\backslash y_1$	$z_2\backslash(S/z_1)$	t_2	m_1/m_2	o_2	v_2	$(w_1\backslash S)/w_2$
Marie	np	x_1	$y_1\backslash S$	z_1	$t_1\backslash S$	m_2	$o_2\backslash(o_1\backslash S)$	$v_2\backslash v_1$	w_2

TABLE 6.1: Unification of variable categories and fixed (learned) categories

Definition 6.3. Pattern matching:

Let t be a category which consists of at least one variable category and possibly some fixed ones, and let L be a set of fixed categories, we say t is matchable with set L, if there exist a substitution of t which is a category in L or a sub-category in L. For

[5] The Catalan(n) is given in terms of binomial coefficients by following formula:

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k} \qquad \text{for } n \geq 0.$$

instance, if we define $L = \{np, S, S/(np\backslash S)\}$ and $t = np\backslash x_1$, then, since the substitution $x_1 := S$ exists and $np\backslash S$ is a sub-category of $S/(np\backslash S)$ in L, we can say that t is matchable with L[6].

Definition 6.4. Derivation of variable categories:

Let X, Y and Z be categories with at least one primitive variable and A/B[7] a category with no primitive variable category (=fixed category). We can define, the derived result of X and A/B as $Y[X := Y/(A/B)]$; and the derived result of A/B and X as $Y[X := (A/B)\backslash Y]$ or $A[X := B]$[8]; and finally the derived results of X and Y as $Z[Y := (X\backslash Z)]$ or $Z[X := (Z/Y)]$. The notation $t := t'$ means substitution of t with t'.

Table 6.1 illustrates fixed learned categories (the second column from left), and eight columns showing the variable categories (labeled from I to VIII) which are the yields of derivation trees in the figure 6.1. We want to find one column, let say n (between I to VIII), such that every line k of the fixed learned categories unifies with the k line of the n column. It can be clearly observed that there is only one column (among eight possible cases) which is unifiable with the fixed learned categories and it is (VIII). By simple substitution we can observe that $w_1 = np$, $w_2 = np$, and since $(w_1\backslash S)/w_2$ matches with $(np\backslash S)/np$. One can observe how the rest seven cases fail in the unification phase. So, we can claim that structural tree (VIII) is unifiable with the lexicon and more important, the substitutions of $w_1 = np$ and $w_2 = np$ in the structural tree (VIII) yield its final derived tree.

It is worth mentioning that the technique described above can distinguish the same word used with different categories, and we will also see how the technique we propose can potentially be generalized to k valued AB grammars in the conclusion section. Having these basic notations will allow us to start describing our techniques for dealing with uncompleted utterances.

6.2.2 Unification Technique of RG Algorithm

The unification technique which is used here is introduced for the first time in learning rigid AB grammars algorithm [BP90] and it has been re-studied and extended by some researchers as well [Kan98; Bon00; BR14]. We are using almost the same technique, but with a new purpose which is dealing with incomplete utterances with one missing category that to our knowledge is a new field to experience. Since we have explained the essential notions, we just focus on some examples to highlight the mechanism of unification in this section.

Let us consider the following lexicon which is adapted for our illustration purpose, and it is assumed that the lexicon is already learned from the positive sentences in corpus[9]. Now, we consider the following examples where a category is missing.

[6] As might be noted, pattern matching here is very similar to unification. We have adopted this notation to make distinction between unification of two categories, and one category with some external references.

[7] Notice that the same procedure with slight modification can be straightforwardly formulated for the fixed category $A\backslash B$.

[8] This new possible combination is introduced for the first time in this dissertation and it was absent in the original paper [Mir].

[9] See [MR12b, Section 1.6.3] for technical details of learning AB rigid grammar; although, this is not essential for grasping the content of this section.

One missing determiner in the example 6.2 and one missing preposition in the example 6.3:

Lexicon:

le , la : np/n	dans : $(s\backslash s)/np$	poisson , mer: n	nage : $np\backslash S$	vite : $(np\backslash S)\backslash(np\backslash S)$

Example 6.2. * Poisson nage vite.

Example 6.3. * Le poisson nage vite la mer.

Let us start with the example 6.2. As it is illustrated in the figure 6.2 we have illustrated only three cases. (Just note that without knowing the categories of words there would be 160 (= $catalan(3) * 2^3 * (3+1)$) possible functor-argument tree structures that can be associated to the example 6.2. We can unify the gained variable categories associated with the words in our structural trees with the one we have in our lexicon (figure 6.2).

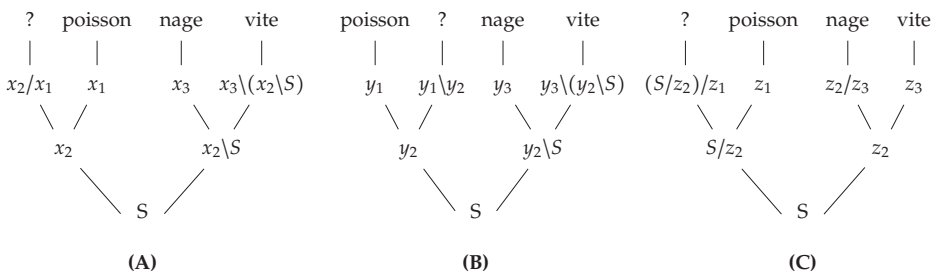


FIGURE 6.2: Three possible structural trees for the example 6.2

Words	Learned categories	A	B	C
poisson	n	x_1	y_1	z_1
nage	$np\backslash S$	x_3	y_3	z_2/z_3
vite	$(np\backslash S)\backslash(np\backslash S)$	$x_3\backslash(x_2\backslash S)$	$y_3\backslash(y_2\backslash S)$	z_3
?	–	x_2/x_1	$y_1\backslash y_2$	$(S/z_2)/z_1$

TABLE 6.2: Unification of variable types and learned types

We can observe that structural tree (A) is unifiable and the solutions are as follows: $x_1 = n$, $x_2 = np$ and $x_3 = np\backslash S$; while the missed category can be gained by substitution of all occurrences of x_i in variable category x_2/x_1 by its relevant fixed category which yields np/n in our case. The suggested category with unification technique on this tree can only be licensed if it exists in our lexicon (or some valid external lexicon) which is the case. Notice that np/n corresponds to determiner category.

Structural tree (B) is unifiable as well, and the unification technique yields $n\backslash np$ as a potential solution for the missing category with the same analysis. Since this solution does not exist in our lexicon it cannot be accepted. Structural tree (C) is not unifiable and it is rejected without further analysis. We can analyze the example 6.3 with the same technique. The problem is a large number of the possible combinations of its structural tree. We will see a treatment for this issue in the next section.

6.2.3 AB grammars, Unification, and Dynamic Programming

As we saw in the previous section, for (unknown) n sequence of words the possible structural trees are (= $catalan(n-1) * 2^{n-1}$) and also the number of positions for a missing category is $n+1$.[10] The final number of different combinations of a n sequence of words with one missing category is = $catalan(n) * 2^n * (n+1)$ which grows faster than exponential rate [Knu73]. Actually, we do not need to explore all the possible structural trees since we know that some of our categories are fixed. In this section, we illustrate how the problem in the example 6.3 can be tackled with dynamic bottom-up parsing.

The idea behind our solution is to try all the (n+1) positions for the missing category and avoiding searching all the possible structural trees by early pattern matching and unification technique of RG algorithm using a modified version of CYK dynamic programming technique [11]. With this strategy we can save each step results for making decision as we go further. As we will see, this leads us to an efficient computational complexity of $O(n^4)$ with n number of words. The table 6.3 is one of the possible cases (out of eight) for parsing the example 6.3 assuming that x_1 (=missing category) is between 'vite' and 'la'. We start parsing step-by-step to illustrate how it works. [12]

1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 6.3: Step (1) of parsing with unification for the example 6.3

The table 6.3 shows the initial phase of parsing, in which the fixed rigid learned categories are assigned plus x_1, a variable category, which is supposed to be specified.[13]

2	np	–	$np\backslash S$	? $x_2[x_1 := ((np\backslash S)\backslash(np\backslash S))\backslash x_2]$	? $x_3[x_1 := x_3/(np/n)]$	np	
1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 6.4: Step (2) of parsing with unification for the example 6.3

The table 6.4 shows step 2, i.e. all possible derivations of two sequential categories. Since, we have the variable categories, we can have the flexibility of working with unification technique. For example, variables x_2 and x_3 would be a possible derivation if x_1 be substituted with $x_3/(np/n)$ and $((np\backslash S)\backslash(np\backslash S))\backslash x_2$ respectively.

[10] For n number of words with one missing category, we will have $n-1$ potential solutions between the words, one in the beginning and another one in the end.

[11] See appendix for the pseudo-code of the algorithm

[12] We have used bottom-up representation which is not very intuitive. This choice is made since it is in alignment with chart representation in dynamic programming.

[13] Note that, in principle, we should assume all the possible positions of the unknown missing word which is 7 in this case. We have shown only one of the cases that leads to a solution. Although, our proposed algorithm would consider all the cases.

Notice that we have included the question mark(=?) in the cells of x_2 and x_3 to signify that we have not performed the pattern matching yet. Our algorithm benefits from early pattern matching of variable categories with the lexicon or any external reference that provides categories[14]. Note that all the possible substitutions for x_1, namely $((np\backslash S)\backslash(np\backslash S))\backslash x_2$ and $x_3/(np/n)$ fail in pattern matching phase; since, there is no fixed type category (and even none subcategory) in our lexicon that can be matched with them. Early matching is an efficient technique since it prevents unnecessary computation. As depicted in step (3) in table 6.5 all possibilities for x_1 (in row 2) are ruled out.

3	S	–	? $x_3[x_1 := (np\backslash S)/x_3]$	–	$x_2[x_1 := x_2/np]$		
2	np	–	$np\backslash S$	–	–	np	
1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 6.5: Step (3) of parsing with unification for the example 6.3

Again, we will follow the same procedure that we practiced for step 3 in table 6.5. In this step, all the possible derivations of three possible sequential categories are shown in each cell. The variable category $(np\backslash S)/x_5$ does not match with any fixed category in our lexicon, so, it will be ruled out. Although, x_6/np matches with some categories in our lexicon so we will keep it as we go to the next step. We will continue the same procedure until we will reach to step 7 which is the last one. Table 6.6 shows the last step.

7	$x_5[x_3 := x_5/np]$ $x_4[x_2 := S\backslash x_4]$						
6	–	–					
5	$x_3[x_1 := S\backslash x_3]$	–	–				
4	S	–	–	–			
3	S	–	–	–	$x_2[x_1 := x_2/np]$		
2	np	–	$np\backslash S$	–	–	np	
1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 6.6: Last step of parsing with unification for the example 6.3

In table 6.6, all the early pattern matching is finished, and we should perform the unification. We can find two following results:

- In row 7, what we expect to see is S, so, $x_5 = S$. By substitution we have $x_3 = S/np$, so, $x_1 = S\backslash(S/np)$, since, this category does not exist in our lexicon it is ruled out.
- The same as above, in row 7, what we expect to see is S. By substitution we have $x_4 = S\backslash S$, so, $x_1 = (S\backslash S)/np$, since, this category exists in our lexicon it accepted as a potential answer, and this is what we were initially looking for, a determiner.

[14]Until now, Moot's Grail [Moo10] provides around 900 categories for the French language. This can be used for pattern-matching in this phase as the external reference.

In this section, we informally analyzed how our modification to CYK algorithm can be useful in finding a missing category using early pattern matching and unification in learning RG algorithm.

6.2.4 Algorithm A

This subsection illustrates the algorithm with $O(n^4)$ time complexity. This algorithm, as we stated before, is designed for finding a missing category in an incomplete utterance by using unification technique as when learning categorial grammars, and dynamic programming as in Cocke–Younger–Kasami algorithm.

```
input : An array 'words' and an array 'lexlist' of the categories
output: An array [(pos,cat)] of the pairs indicating position and categories of
        the missing words

result ← [ ];
n ← length(words);
for t ← 1 to n + 1 do
    k ← 1;
    ylist ← null;
    n_words = insert(words, x[k], t);
    // initializes the first row;
    for r ← 1 to n + 1 do
        p [1, r, 1] = lex(n_words[r], null, null)
    end
    for i ← 2 to n + 1 do
        for j ← 1 to n − i + 2 do
            for m ← 1 to i − 1 do
                // performs early pattern matching and unification;
                xlist = unify(p[m, j, 1], p[i − m, j + m, 1], lexlist, k) ;
                if xlist != fail then
                    ylist = ylist + xlist ;
                    assign(p[i, j], xlist) ;
                    else p[i,j,1]=null
                end
            end
        end
    end
    // substitutes the variables with its content;
    for v ← k to 1 step −1 do
        t2 ← snd(ylist(v));
        t3 ← trd(ylist(v));
        if is_fixed(t3) then
            substitute_all(ylist, t2, t3)
        end
    end
    // selects all the appropriate candidates for missing categories;
    for w ← 1 to len(ylist) do
        t2 ← snd(ylist(v));
        t3 ← trd(ylist(v));
        if (is_fixed(t3) and t2=x[1] and is_in_lexicon(lexlist,t3)) then
            result = result + [(t, t3)]
        end
    end
    return (result)
end
```

Algorithm 1: An algorithm for finding one missing

6.2.5 Limitations of Algorithm A

One of the obvious limitations in our algorithm is that it only works with one missing category with $O(n^4)$ time complexity. Finding k fixed number of the missing categories within n words might suggest $O(n^{3+k})$ time complexity. But, this is not the

whole story, since the k number of missing words would bring the space complexity and it grows with $Catalan(k)$ rate which is even faster than exponential rate. All in all, we should say, our solution for k fixed number of the missing category would have a plausible time complexity but not a good space complexity as k increases.

6.3 Sentence Completion: Algorithm B

The algorithm that we introduce in this section is a procedure for resolving missing categories in a given ungrammatical utterance along with suggesting some fixations. We will show how Constraint Handling Rules [Frü95; Frü98] can be suitably exploited for this purpose. The main theme in this section is to provide mathematical proof to show that our algorithm works computationally efficient. The study aims to be an extension for finding more than one missing category in an incomplete sentence at the cost of increasing the complexity of time to an exponential rate.

6.3.1 Syntax and Semantics of Constraint Handling Rules

Constraint Handling Rules (=CHR) is a declarative constraint logic programming language. It is a rule-based language that works with constraints. The concrete syntax of CHR depends on the language in which CHR is embedded. Since Prolog has been acted as the host language in the most known implementations, we have adopted the Prolog terminologies in describing and representing the concrete syntax of CHR. So, we can represent constraint in CHR as terms in Prolog. Prolog's terms are of four types: atoms, numbers, variables, and complex terms (or structures). Complex terms are built out of a functor followed by a sequence of arguments. So, we can understand a constraint in CHR as a form of an atom or a predicate with some arguments.

CHR consists of guarded rules that manipulate the query(goal). Goals can be stated in Prolog as the multi-sets of constraints separated by the comma. The constraints in a goal are processed from left to right. We call an active constraint a constraint that is under checking process for applicability of the rules. We say a rule is fired if its heads matches and its guards are successfully checked. Based on the kind of rule (as described below) different strategies can be followed, such as: keeping constraints in the constraint store(=data structure for CHR), removing it, etc.

Having these basic notions, we can take a look at CHR rules. They are categorized into three types. As follows, syntax and semantic of three types of CHR rules will be described:

Simplification Rules

$$head_1, \ldots, head_n \Longleftrightarrow guard_1, \ldots, guard_m | body_1, \ldots, body_k$$

If a query matches with the heads of a simplification rule, and the guards hold too, then, simplification rules fire, and as the consequence of that it rewrite the $head_1, \ldots, head_n$ into the $body_1, \ldots, body_m$.

Propagation Rules

$$head_1, \ldots, head_n \Longrightarrow guard_1, \ldots, guard_m | body_1, \ldots, body_k$$

For a propagation rule to fire, the query must match with the heads, and the guards must hold true. Rules add the constraints in the body to the constraint store, without removing the heads.

Simpagation Rules

$$head_1, \ldots, head_\ell \backslash head_{\ell+1}, \ldots, head_n \iff guard_1, \ldots, guard_m | body_1, \ldots, body_k$$

For a simpagation rule to fire, the query must match all the rules with the heads, and the guards must hold true. The ℓ constraints before the \ are kept, as a in a propagation rule; the remaining $n - \ell$ constraints are removed.

6.3.2 Converting AB Grammar to CFG in Chomsky Normal Form

Before moving on, we introduce a very straightforward procedure for converting our syntactic analysis in AB Grammar form to Context Free Grammar in Chomsky Normal Form. This conversion is essential since our CHR rules work in CFG format. The following proposition shows the strong equivalence between the two formalisms and also provides a procedure for converting AB Grammar to CFG in Chomsky Normal Form. [MR12b, P.8]:

Proposition: Every AB grammar is strongly equivalent to a CFG in Chomsky normal form.

Proof. Let G be the CFG defined by:

- Terminals T are the words of the AB grammar.
- Non Terminals NT are all the subtypes of the types appearing in the lexicon of the AB grammar— a type is considered to be a subtype of itself.
- The production rules are of two kinds:

– $X \to a$ whenever $X \in Lex(a)$

– $X \to (X/Z)Z$ and $X \to Z(Z\backslash X)$ for all $X, Z \in NT$ —keep in mind that from the CFG viewpoint $(Z\backslash X)$ and (X/Z) are both non terminal symbols.

This defines a CFG because the lexicon is finite, so there are only finitely many subtypes of types in the lexicon, hence finitely many production rules. The derivation trees in both formalisms are isomorphic.

6.3.3 Algorithm B: Syntax of the Constraint Rules

In this section, a high-level informal description of the algorithm for finding the missing syntactic categories of an ungrammatical utterance is described. We will describe eventually some formal aspects of the algorithm with its properties and mathematical proofs.

Let G' be an AB Grammar, and let G be a CFG in Chomsky Normal Form which is transformed from G' by the procedure described in 6.3.2, and let u be a sequence of syntactical non-terminal as input, and $\vec{u} = [A_1, \ldots, A_n]$ be the multiplicity vector of u which is gained by counting the number of each category in u. Let $c(A_1, \ldots, A_n)$

be a constraint–in CHR sense– that is built from $\vec{u}$ by copying $\vec{u}$ elements as the arguments of predicate c. Any CHR program works with some rules which are stated in the knowledge-base. We will build these rules from a G by the following procedure; as it is indicated we have two type of descending rules and three types of ascending rules.

CFG Rule	Corresponding CHR Rule	Rule Type
$A_i \rightarrow A_i A_j$ or $A_i \rightarrow A_j A_i$	$c(A_1,\ldots,A_n) \Rightarrow A_j > 0\,, S \geq 1 \mid A_j = A_j - 1, c(A_1,\ldots,A_n).$	Descending (I)
$A_i \rightarrow A_i A_i$	$c(A_1,\ldots,A_n) \Rightarrow A_i > 0\,, S \geq 1 \mid A_i = A_i - 1, c(A_1,\ldots,A_n).$	Descending (II)
$A_i \rightarrow A_j A_k$	$c(A_1,\ldots,A_n) \Rightarrow A_j > 0 \vee A_k > 0 \mid A_j = A_j - 1,\ A_k = A_k - 1,\ A_i = A_i + 1\,,\ c(A_1,\ldots,A_n).$	Ascending (I)
$A_i \rightarrow A_j A_j$	$c(A_1,\ldots,A_n) \Rightarrow A_j > 0 \mid A_j = A_j - 2,\ A_i = A_i + 1\,,\ c(A_1,\ldots,A_n).$	Ascending (II)
$A_i \rightarrow A_i A_j$ or $A_i \rightarrow A_j A_i$	$c(A_1,\ldots,A_n) \Rightarrow A_j > 0\,, S = 0 \mid A_j = A_j - 1,\ A_i = A_i + 1,\ c(A_1,\ldots,A_n), e(0,\ldots,0,-1^{A_i},0,\ldots,0).$	Ascending (III)
$A_i \rightarrow a$	Not Applicable	Not Applicable

The ascending rules of type (III) generate new extra predicate which is called e in the body of CHR rule. All of the arguments of e is zero except the ith component of e which is '-1'. In order to avoid the unwanted complexity which is naturally forced by introducing this kind of extra CHR constraint(=predicate), we need to collect them through simple merging operation techniques. This is the simple philosophy behind introducing a new general type of rules that called collective. As we will see introducing collective rules will have no significant mathematical value for us, and it can be considered just as a computational trick to make things mathematically simple. (Recall that we need to provide mathematical proofs to show that our algorithm behave computationally well)

Name	CHR Rule	Type
General Rule	$c(A_1,\ldots,A_n), e(B_1,\ldots,B_n) \Leftrightarrow S \geq 1 \mid C_i = A_i + B_i\ (for\ 1 \leq i \leq n),\ c(C_1,\ldots,C_n).$	Collective

To put things in the big picture, we can restate that, we have u as our goal, and we have a number of CHR(G) rules built from G. We expect to have a sequence of rules applications that will terminate. (or we can say in CHR sense that no rule will be fired). After termination, we have a vector (or a constraint in CHR sense) that is our solution. In other words, if the final constraint ends up with $(1^S, 0, \ldots, 0)$ it indicates to no missing categories; if it ends up with categories with negative numbers it indicates the number of missing categories and categories with a positive number indicates extra ones.

We have not yet discussed how the parsed trees can be gained as the side-effect

of running CHR(G) rules. As we will see, properties (I) and (J) and their relevant proofs provide a precise procedure for such a phenomenon. For now, we just provide two examples to see how CHR(G) rules can work in real running examples.

Case Study:

Here, the print-out of two queries and their results is illustrated. It is a Prolog/CHR implementation of some toy examples with a given following CFG[15] with relative clauses. The first example is a grammatical sentence while the second one is ungrammatical.

CNF:
np → pn.
np → det,cn.
np′ → rc, vp.
np → np, np′.
vp → v[iv].
vp → v[tv], np.
s → np, vp.

Example 6.4. A magician who helped a girl laughs.

Input: [det, cn, rc, v(tv), det, cn, v(iv)].

Goal: c[(cn,2), (det,2), (pn,0), (rc,1), (v(iv),1), (v(tv),1), (np,0), (np′,0), (vp,0), (s,0)]

Chain of Rules Application:

c[(cn,1), (det,1), (pn,0), (rc,1), (v(iv),1), (v(tv),1), (np,1), (np′,0), (vp,0), (s,0),Ascending]
*Remark: det and cn are decreased by 1 and np is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,1), (v(iv),1), (v(tv),1), (np,2), (np′,0), (vp,0), (s,0),Ascending]
*Remark: det and cn are decreased by 1 and np is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,1), (v(iv),0), (v(tv),1), (np,2), (np′,0), (vp,1), (s,0),Ascending]
*Remark: $v(iv)$ is decreased by 1 and vp is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,1), (v(iv),0), (v(tv),0), (np,1), (np′,0), (vp,2), (s,0),Ascending]
*Remark: $v(tv)$ and np are decreased by 1 and vp is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,1), (v(iv),0), (v(tv),0), (np,0), (np′,0), (vp,1), (s,1),Ascending]
*Remark: np and v are decreased by 1 and S is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),0), (np,0), (np′,1), (vp,0), (s,1),Ascending]
*Remark: rc, vp are decreased by 1 and np' is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),0), (np,0), (np′,0), (vp,0), (s,1),Descending]
*Remark: np' is decreased by 1.

We can recognize the pattern $(1^s, 0, \ldots, 0)$ terminated with the last constraint. This

[15]Notice that in order to increase the readability of rules the occurrence number of each category is followed together with the label of its category in $(Cat, \#Cat)$ format.

indicates to a corrected grammatical sentence.

Example 6.5. *A girl helped the child admires the magician.

Before going through this example we should keep in mind the two plausible candidates as follows:

S_1: A girl [who] helped the child admires the magician.
S_2: A girl helped the child [who] admires the magician.

Now, we can take a look to the result of the algorithm.

Input: [det, cn, v(tv), det, cn, v(tv), det, cn].
Goal: c[(cn,3), (det,3), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,0), (vp,0), (s,0)]

Chain of Rules Application:

c[(cn,2), (det,2), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,1), (np′,0), (vp,0), (s,0),Ascending]
*Remark: det and cn are decreased by 1 and np is increased by 1.

c[(cn,1), (det,1), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,2), (np′,0), (vp,0), (s,0),Ascending]
*Remark: det and cn are decreased by 1 and np is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,3), (np′,0), (vp,0), (s,0),Ascending]
*Remark: det and cn are decreased by 1 and np is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),1), (np,2), (np′,0), (vp,1), (s,0),Ascending]
*Remark: $v(tv)$ and np are decreased by 1 and vp is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),0), (np,1), (np′,0), (vp,2), (s,0),Ascending]
*Remark: $v(tv)$ and np are decreased by 1 and vp is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),0), (np,0), (np′,0), (vp,1), (s,1),Ascending]
*Remark: np and vp are decreased by 1 and S is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,-1), (v(iv),0), (v(tv),0), (np,0), (np′,1), (vp,0), (s,1),Ascending]
*Remark: rc and vp are decreased by 1 and np' is increased by 1.

c[(cn,0), (det,0), (pn,0), (rc,-1), (v(iv),0), (v(tv),0), (np,0), (np′,0), (vp,0), (s,1),Descending]
*Remark: np' is decreased by 1.

We can see that relative clause number is '-1'. As explained, this indicates a missing category of type relative clause. The question of how to put this missing category in two possible places in the sentence is addressed in the property (I) and (J) of the algorithm which is described in the appendix A.

6.3.4 Algorithm B: Semantics of the Constraint Rules

Let $NT = \{A_1, \ldots, A_n\}$ be a set of non-terminals, in which, n is the size of NT. Given a sequence u of NT^*, let us call the multiplicity vector of u the vector $\vec{u} = (NoA_1(u), \ldots, NoA_n(u))$, where $NoA_i(u)$, is the number of occurrence of the

non-terminal A_i in u.

Let G' be an AB Grammar, and let G be a CFG in Chomsky Normal Form which is transformed from G' by the procedure described in 6.3.2. Now, let us define $CHR(G)$, a specific CHR program, consisting of a sequence of CHR re-write rules built from G and u. Here are the meta-rules for turning production rules of G into CHR re-write rules.

CFG Rule	Meaning of Corresponding CHR Rule	Rule Name
$A_i \rightarrow A_iA_j$ or $A_i \rightarrow A_jA_i$	If $(\pi_j(\vec{w}) > 0$ *and* $S \geq 1)$ then $decrease_by_one(\vec{w}, j)$.	Descending (I)
$A_i \rightarrow A_iA_i$	If $(\pi_i(\vec{w}) > 0$ *and* $S \geq 1)$ $decrease_by_one(\vec{w}, i)$.	Descending (II)
$A_i \rightarrow A_jA_k$	If $(\pi_j(\vec{w}) > 0$ *or* $\pi_k(\vec{w} > 0))$ then $increase_by_one(\vec{w}, i)$, $decrease_by_one(\vec{w}, j)$ and $decrease_by_one(\vec{w}, k)$.	Ascending (I)
$A_i \rightarrow A_jA_j$	If $(\pi_j(\vec{w}) > 0)$ then $increase_by_one(\vec{w}, i)$ $decrease_by_two(\vec{w}, j)$	Ascending (II)
$A_i \rightarrow A_iA_j$ or $A_i \rightarrow A_jA_i$ $A_i \rightarrow A_jA_i$	If $(\pi_j(\vec{w}) > 0$ *and* $S = 1)$ then $increase_by_one(\vec{w}, i)$. then $decrease_by_one(\vec{w}, j)$.	Ascending (III)
$A_i \rightarrow a$	Not Applicable	Not Applicable

Where i, j and k are assumed to be distinct; $\pi_i(\vec{w})$ is the value of ith component of $\vec{w}$; and $increase_by_one(\vec{w}, i)$ increases by one the value of ith component of $\vec{w}$. Before moving on, we adopts following definitions which we will extensively use in our mathematical proofs in the appendix A:

Definition 6.5. Conversion:

A multiplicity vector $\vec{u}$ converts to $\vec{v}$ ($\vec{u} \rightarrow \vec{v}$), under a given CHR(G), if there exists a CHR rule such that the head of rule matches with $\vec{u}$ and also the guard of rule holds. As the result of that, the CHR rule will fire, and $\vec{v}$ would be gained. We can see this process as a simple replacement between two vectors by an instruction that is coded in a CHR rule.

Definition 6.6. Reduction:

A given multiplicity vector $\vec{u}$ reduces to $\vec{v}$ when there is a sequence of conversions from $\vec{u}$ to $\vec{v}$–under a given CHR(G)– that is a sequence $\vec{u} = t_0, t_1, \ldots, t_n = \vec{v}$. We write $\vec{u} \rightsquigarrow \vec{v}$ for $\vec{u}$ reduces to $\vec{u}$.

Definition 6.7. Saturation:

A multiplicity vector $\vec{v}$ with no possible conversion, under a given CHR(G), is called saturated.

Definition 6.8. Candidate:

A multiplicity vector $\vec{v}$ is called a candidate, if the number of occurrence of starting symbol S is greater than or equal to 1.

Definition 6.9. Degree of multiplicity vector:

The number(=degree) of a multiplicity vector $\vec{v}$ is the sum of all its positive arguments without counting the occurrence of start symbol (S). We write $d(\vec{v})$ for the degree of multiplicity vector $\vec{v}$ which is obviously greater or equal to zero.

Definition 6.10. Reachability:

In a given CFG G, a non-terminal X is reachable to Y in h steps, if and only if: $\exists\, u_1, \cdots, u_h, \in NT^*\ \exists\, \alpha, \beta \in NT^*$: $Y = u_0 \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_h = \alpha X \beta$.

Definition 6.11. Proper CFG:

A context-free grammar is said to be proper [Hop79; Nij80], if it has:

- no unreachable symbols: $\forall X \in NT\ \exists \alpha, \beta \in (NT \cup T)^* : S \overset{*}{\Rightarrow} \alpha X \beta$
- no unproductive symbols: $\forall X \in NT\ \exists w \in T^* : X \overset{*}{\Rightarrow} w$
- no e-productions: $\neg \exists X \in NT : (X, \varepsilon) \in R$
- no cycles: $\neg \exists X \in NT : X \overset{+}{\Rightarrow} X$

We assume that all CFGs in our research are proper CFG, even when it is not explicitly declared.

Definition 6.12. Proper CHR:

A set of CHR rewrite rules is said to be proper, if it is prioritized in the following order, namely, written in our knowledge-base from top to bottom:

- Collective Rule
- Descending Rules of Type I
- Descending Rules of Type II
- Ascending Rules of Type I
- Ascending Rules of Type II
- Ascending Rules of Type III

All the rules of the same type which is mentioned above should be internally prioritized in terms of reachability number of the non-terminals in L.H.S of the corresponding CFG rule to the start symbol S, namely, all rules should be sorted from high reachability numbers (of L.H.S non-terminals to start symbol S) to the low numbers appearing respectively in knowledge-base in top to bottom order. Again, here, we assume that all CHRs in our research are proper CHR, even when it is not explicitly declared.

6.3.5 Properties of the Algorithm B

The algorithm that we have provided has the following good properties. The mathematical proofs for each of the properties here are provided in the appendix A.

Property (A)
For every h, for every partial derivation tree t with height h, yield w and root r of grammar G, such that, each leave label is of NT, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

Property (B)
For every partial derivation tree t with yield w and root r of grammar G, such that, each leave label is of NT, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

Property (C)
For every sequence w of NT^* produced by grammar G from r, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

Property (D)
(A) If a multiplicity vector $\vec{t_i}$ converts to a vector $\vec{t_{i+1}}$ under only the descending rules of a given CHR(G), then, $d(\vec{t_{i+1}}) < d(\vec{t_i})$.

(B) If a multiplicity vector $\vec{t_i}$ converts to a vector $\vec{t_{i+1}}$ under only the ascending rules of a given CHR(G), then, $d(\vec{t_{i+1}}) \leq d(\vec{t_i})$.

Property (E)
A multiplicity vector $\vec{u}$ with zero degree has no applicable CHR rule.

Corollary: A reduction from a multiplicity vector $\vec{u}$ to a vector $\vec{v}$ with zero degree is terminated, since, $\vec{v}$ can not be converted to any vector.

Property (F)
For every $h \geq 1$, for every non-terminal x with positive number of occurrences in $\vec{u}$, and being reachable to a category y in h steps in a given proper CFG, G, there exists a sequence of reduction $\vec{u} = t_h, t_{h-1}, \ldots, t_0 = \vec{v}$ with length h, under a CHR(G), such that, the number of y in $\vec{v}$ is increased by one.

Property (G)
There is a reduction sequence from any multiplicity vector $\vec{u}$ to a candidate vector $\vec{v}$ under a CHR(G) with G as proper CFG while $d(\vec{v}) < d(\vec{u})$.

Property (H)
For every h, under a proper CHR(G) and proper G, all possible sequence of reductions of $\vec{u}$ of degree $h = d(\vec{u})$ terminate.

Property (I)
If $\vec{u} \rightsquigarrow \vec{t_h}$ ($\vec{u} = t_0, t_1, \ldots, t_h$), and $c_1, \ldots, c_h$ be the sequence of CHR(G) rules such that (for $i = 0, 1, \ldots, h-1$) t_{i+1} be conversion of t_i under c_{i+1} CHR(G) rule, then:

(A) There exists a procedure on the basis of the sequence $c_1, \ldots, c_h$ for building corrected parsed tree(s) of u.

(B) The elements in multiplicity vector t_h suggests the number of missing categories if it is negative; not-yet-used categories if be positive; and finally, zero indicates no-suggestion-so-far.

Property (J)
If $\vec{u} \rightsquigarrow \vec{t_h}$ ($\vec{u} = t_0, t_1, \ldots, t_h = \vec{v}$), and $\vec{v}$ be a saturated candidate and $c_1, \ldots, c_h$ be the sequence of CHR(G) rules such that (for $i = 0, 1, \ldots, h-1$) t_{i+1} be conversion of t_i under c_{i+1} CHR(G) rule, then:

(A) There exists a procedure on the basis of the sequence $c_1, \ldots, c_h$ for building corrected parsed tree(s) of u.

(B) The elements in multiplicity vector t_h suggests the number of missing categories if it is negative; extra categories if it is positive; and finally, zero indicates no-suggestion.

6.3.6 Limitations of Algorithm B

We have verified our algorithm for subjective/objective relative clauses by implementing it with some toy examples in Prolog/CHR programming languages. The implementation shows some good properties such as terminations and soundness of the solutions for this particular test. However, we should admit that our algorithm is not good in terms of its time complexity. The problem of finding n number of missing categories by using our algorithm would have some solutions at the cost of facing the exponential complexity. This basically happens due to this fact that we should search all the possible search spaces for finding the solutions that suggest the lesser number of missing words. Thus, our solution does not have the property of a wide-coverage applicability.

6.4 Ranking Interpretations by Means of Categorial Proof Nets

As discussed in the introduction, one of the main reason for choosing *Categorial Grammars* is its excellent syntax-semantic interface. With the procedures described in the chapter 2 we can gain the possible interpretations of incomplete sentences by constructing different categorial proof nets and reading off the semantic readings associated with them. Now, we should look for a computational model to rank these possible readings for an incomplete sentence, and of course, we would like to support our rankings in accordance to the psycholinguistic theories on the human linguistic performances. A naive attempt would be to adopt theories introduced in the chapters 3 and 5, namely Incomplete Dependency Theory and Dependency Locality Theory on the categorial proof nets. This would be in general a good start point, although, as we will see, is not efficient for some of the linguistic phenomena. In the remaining part of this chapter, after integration of our algorithms with the IDT-based and DLT-based complexity profiling, we will focus on two new things: (i) computational modeling for syntactic grammaticality judgment limited to the sentences with missing categories. (ii) we will also investigate activation-based linguistic preferences, and our new approach to model this phenomenon by means of categorial proof nets.

6.4.1 Quantifying Preferences with Distance-based Theories

Now, we want to see how the outcome of our algorithms as the potential interpretations can be ranked in alignment with the IDT-based and DLT-based complexity profiling. Let us consider the following example as our toy example [16]:

Example 6.6. *Every child fell cried.

6.6a. Every child fell [and] cried.
$\forall x(Child(x) \rightarrow Fell(x) \wedge Cried(x))$

6.6b. Every child [who] fell cried.
$\forall x(Child(x) \wedge Fell(x) \rightarrow Cried(x))$

As one can observe, there are two candidates for fixing 6.6. The first one is to use a conjunction *and* between two verbs as in the 6.6a and the second one is to use a subjective relative pronoun *who* to fix the sentence as it is shown in 6.6b. The two candidates do not have the same meaning at all. As in the 6.6a, we can not imagine a child who cried and did not fall. However, in 6.6b, we can imagine a child who is crying and she has not fallen, namely, crying for other reasons than falling.

In the first place, we will show how to fix the incomplete sentence 6.6 by performing the algorithm (a) with the following lexicon: (we will see that the expected results would be easily gained)

Lexicon:

Every : np/n	child : n	fell: $np\backslash S$	cried: $np\backslash S$	who: $(n\backslash n)/(np\backslash S)$
and: $(np\backslash S)\backslash((np\backslash S)/(np\backslash S))$				

The tables 6.7 and 6.8 illustrates how the algorithm (a) can detect the two possible places and two different categories relevant to the places for the completion of the sentence 6.6. We can find two following results by looking at the table 6.7:

- In row 5, what we expect to see is S, so, $x_5 = S$. By substitution we have $x_2 = (np\backslash S)/(np\backslash S)$, so, $x_1 = (np\backslash S)\backslash((np\backslash S)/(np\backslash S))$.
- With the same procedure as above and starting from $x_6 = S$, we can gain $x_1 = ((np\backslash S)\backslash(np\backslash S))/(np\backslash S)$

Both categories in general scheme can be rewritten as $X\backslash(X/X)$ and $(X\backslash X)/X$ which corresponds with the conjunction *and* as we expected.

5	$x_5[x_4 := np\backslash x_5]$, $x_6[x_5 := np\backslash x_6]$				
4	–	–			
3	S	–	$x_4[x_2 := x_4/(np\backslash S)]$, $x_5[x_3 := (np\backslash S)\backslash x_5]$		
2	np	–	$x_2[x_1 := (np\backslash S)\backslash x_2]$	$x_3[x_1 := x_3/(np\backslash S)]$	
1	np/n	n	$np\backslash S$	x_1	$np\backslash S$
	Every	**child**	**fell**	**?**	**cried**
	1	**2**	**3**	**4**	**5**

TABLE 6.7: Parsing with unification for the example 6.6

[16] As we will see, this approach has it own limitation and we need better solutions. See sub-section 6.4.2 for more details

5	S				
4	$np[x_4 := n]$	–			
3	–	$x_4[x_3 := n\backslash x_4]$	–		
2	np	$x_2[x_1 := n\backslash x_2]$	$x_3[x_1 := x_3/(np\backslash S)]$	–	
1	np/n	n	x_1	$np\backslash S$	$np\backslash S$
	Every	**child**	**?**	**fell**	**cried**
	1	**2**	**3**	**4**	**5**

TABLE 6.8: Parsing with unification for the example 6.6

In the table 6.8, we can also observe that $x_4 = n$. By substitution we have $x_3 = (n\backslash n)$, so, $x_1 = (n\backslash n)/(np\backslash S)$. This would suggest the subjective relative clause *who* as it exists in our lexicon.

Now, it is time to measure the preferences for the two readings by applying the IDT-based and DLT-based complexity profiling technique that we introduced in the chapters 3 and 5. Considering the figure 6.3 which illustrates the relevant categorial proof nets for 6.6a and 6.6b, we can gain the linguistic complexities in the tables 6.9 and 6.10. Both IDT-based and DLT-based approaches show that 6.6b is more complex than 6.6a. Thus, we can truly express that based on both theories, 6.6a would be preferred over 6.6b. We do not have any psycholinguistic evidence for supporting this prediction for the time being. Different readers of this dissertation may prefer one of the readings and we do not want to deny this fact. However, we believe that 6.6b is a general claim while the 6.6b is a more specific claim and it needs a context to be preferred.

6.6a	Every 3	child 2	fell 4	and 2	cried 0
6.6b	Every 3	child 4	who 4	fell 2	cried 0

TABLE 6.9: Calculation of the incomplete dependency number for 6.6a and 6.6b.

6.6a	Every 0	child 0	fell 0	and 1	cried 1
6.6b	Every 0	child 0	who 1	fell 1	cried 3

TABLE 6.10: Calculation of the dependency locality number for 6.6a and 6.6b.

We need to highlight two important things: firstly, both IDT-based and DLT-based complexity profiling techniques predict the preference of one specific meaning over the other one. Secondly, even if some future experimental studies support this prediction, this kind of preference modeling definitely fails if we take into account other well-studied performance phenomena such as activation theory (discussed in the next section). As we stated in the introduction, this evidently shows that non-canonical sentences are way more complex than what we expect, and we need a special kind of treatment for each case in order to capture them in our preferential semantic modeling.

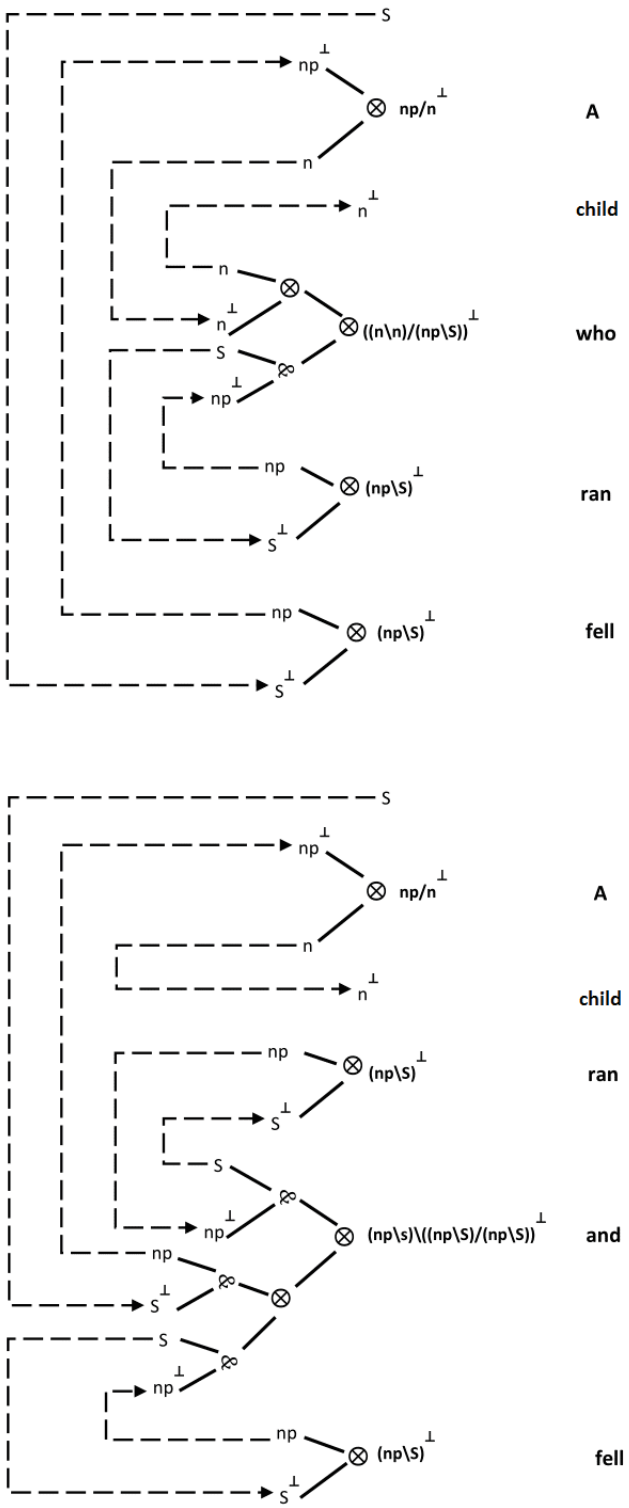


FIGURE 6.3: Proof net analyses for 6.6a and 6.6b.

6.4.2 Quantifying Preferences with Activation Theory

In the previous subsection, we took into account some psycholinguistic distance-based theories such as IDT and DLT for measuring the linguistic complexity of the (possible) interpretations of the incomplete sentences. We investigated some examples regarding the self-center-embedding constructions in the chapter 5. The main interest in these examples is that they would inspire modeling the short-term or working memory. Broadly speaking, a category, let say X, would be temporarily stored in working memory until another category such as $X\backslash S$ necessitates the retrieval of the X for the further integration process. We saw, in great details, how DLT and IDT might be helpful in this regard.

In this section we concentrate on the examples that counterbalance these theories. Incomplete sentences such as 6.7a and 6.7b (taken from [Bla11a; Bla11b]) can reflect this serious lacking.

Example 6.7.

6.7a. *The rat the cat saw died.

6.7b. *The rat the cat briefly saw died.

Experiment shows the preference of 6.7b over 6.7a [Vas+05]. The difficulty that a human performer faces in comprehending 6.7b is lesser comparing to 6.7a , even though, it is lengthier comparing to 6.7a.

The psycholinguistic theory that is responsible for explaining this phenomenon is the *activation theory* [Vas+05; LV05] which is formalized in computational form in the Adaptive Control of Thought-Rational (ACT-R) architecture [And+04]. This theory construes parsing as an efficient series of the guided memory retrievals. We can broadly understand this phenomenon by explaining the step-by-step human sentence processing in our example. In 6.7a, after processing the first NP_1, namely *the rat*, a VP_1 is expected. Again, after processing the second NP_2, namely *the cat*, another VP_2 is expected. These VP_1 and VP_2 which are constructed on the fly, have a base-level activation that decays as the time passes. Since no adverb is intervened between NP_2 and VP_2, the VP_2 (*saw*) would be integrated into the structure which is built incrementally till now. This integration is not straightforward because the reader should spend more time to examine different possibilities which would end up to the connection NP_1, VP_1 and NP_2, VP_2. This would suggest that the expression *the cat saw* should be connected to NP_1 with a proper relative clause. Finally, the expression *the rat [that] the cat saw* would be built. In this scenario, the total retrieval of NP_1 and NP_2 demands more time to what we will explain regarding 6.7b. We assume this retrieval time t. In the case 6.7b, the adverb *briefly* intervenes. A part of the adverb processing involves retrieving the newly created VP_1. This retrieval occurs faster in time $t' < t$. This gives an explanation why 6.7b is processed faster than 6.7a.

Now, let us see what happens if we want to apply the previous metrics, i.e. IDT-based and DLT-based complexity profilings. Firstly, we should apply one of the algorithm (a) or (b) to find the missing category in both sentences.[17] Our general linguistic knowledge suggests to add a word with a relative clause category between *the rat* and *the cat*. As we expect, our algorithm can correctly find the category

[17] We have not illustrated the relevant charting parse, since, it is a straightforward case and we have explained in lengthy details three similar examples in this chapter.

$((n\backslash n)/(np\backslash S))$ as well.

The tables 6.11 and 6.12 show that both IDT-based and DLT-based complexity profilings fails in prediction this phenomenon. As one can observe, the complexity of 6.7b is higher than 6.7a. We described why this is evidently wrong on the basis of the activation theory.

	The	rat	that	the	cat	saw	died	
6.6a	3	4	4	6	5	2	0	
	The	rat	that	the	cat	briefly	saw	died
6.6b	3	4	4	6	5	5	2	0

TABLE 6.11: Calculation of the incomplete dependency number for 6.7a and 6.7b.

	The	rat	that	the	cat	saw	died	
6.6a	0	0	1	0	0	2	4	
	The	rat	that	the	cat	briefly	saw	died
6.6b	0	0	1	0	0	1	2	4

TABLE 6.12: Calculation of the dependency locality number for 6.7a and 6.7b.

Now, we have to do something to integrate this new phenomenon into our model in a consistent way. For sure, we are not going to use the activation-based models [LV05] which use Adaptive Control of Thought-Rational (ACT-R) architecture [And+04]. The reason for this strategy is that this framework does not provide the logical formula that we need for representing the natural language meaning.

For modeling activation theory, we will be able to proceed to so some extent if we model the conditions that trigger the activation phenomenon. With this strategy this is quite possible, and yet plausible, to use the proof nets. In other words, we find the typical linguistic expressions that cause activation. Using categorial grammars aligns with our general strategy as we sketched out in the introduction.

Some studies [Bla11b; Bla11a; Hal01] suggest a good direction for integration activation theory into IDT-based complexity profiling. Based on these studies, structures with more properties— satisfied early in the sentence— are preferred. This is regularly the case with pre-modifiers that always activate the head. This would help us to trigger our model for measuring this phenomenon. We reduce the effect of the activation from our IDT-based complexity profiling as we will see in the following definitions. We will implement a simple idea: **the activator (the adverb for instance in 6.7b) and activated (the VP for instance in 6.7b) words reduce the cognitive-effort for sentence comprehension, since they are more predictable in the context, and the human performance would need less cognitive-effort to do so**. Let us see how this works in the following definitions:

Definitions of the Activation-based Measuring

Definition 6.13. A word w is said to be an **activator word** when it is a pre-modifier that activates the head phrase in an expression. The mentioned head is said to be the

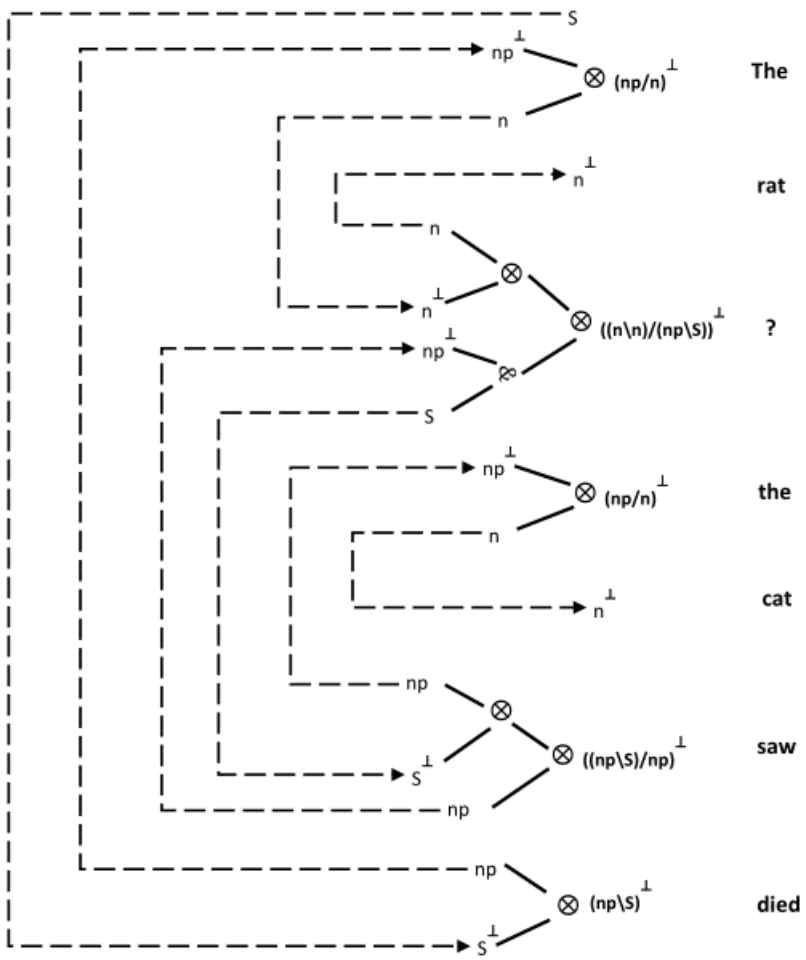


FIGURE 6.4: Proof net analysis for 6.7a.

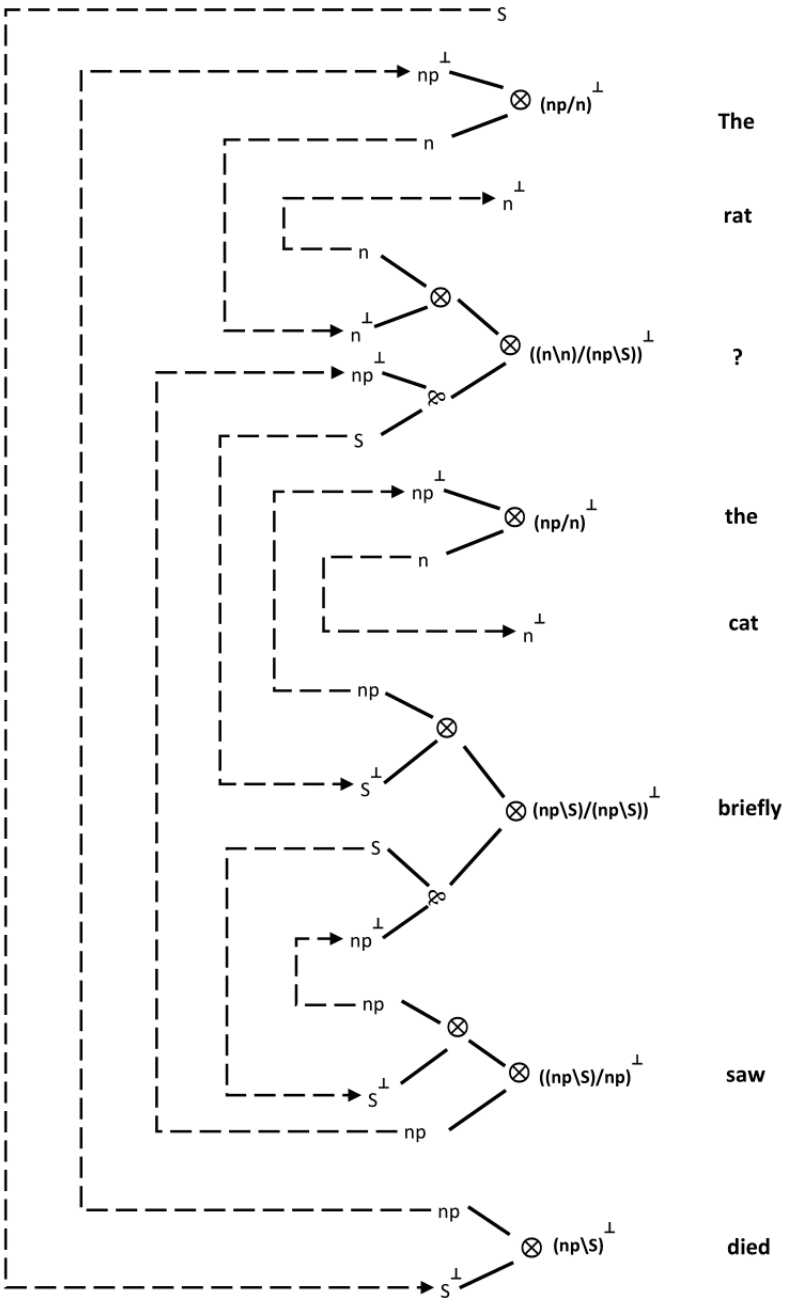


FIGURE 6.5: Proof net analysis for 6.7b.

activated word.

Definition 6.14. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let $c - c'$ be an axiom in π such that $c \in C_i$ and $c' \in C_j$ $(i, j \in [1, n])$. We define the **activation weight** of the axiom $c - c'$ as 0 if $w_i : C_i$ or $w_j : C_j$ is an activator or activated word. Otherwise, the **activation weight** of the axiom $c - c'$ is 1.

Definition 6.15. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i $(i \in [1, n])$. The activation number of C_{i_0} in π, written as $AN_\pi(C_{i_0})$, is the sum of axioms **activation weights** $c - c'$ in π such that $c \in (C_{i_0-m} \cup S)$ $(m \geq 0)$ and $c' \in C_{i_0+n+1}$ $(n \geq 0)$.

Definition 6.16. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. We define the Activation-based linguistic complexity of π, written $f_{idat}(\pi)$ by $f_{idat}(\pi) = (1 + \sum_{i=1}^{n} AN_\pi(c_i))^{-1}$.

Definition 6.17. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is Activation-preferred to π_j whenever $f_{idat}(\pi_i) > f_{idat}(\pi_j)$.

	The	rat	that	the	cat	saw	died	
6.6a	3	4	4	6	5	2	0	
	The	rat	that	the	cat	briefly	saw	died
6.6b	3	4	2	3	2	2	2	0

TABLE 6.13: Calculation of the activation-based complexity measurement for 6.7a and 6.7b.

Example: We can observe that the new metric works well in table 6.13. The total sum of the activation numbers for 6.6b is 18, while the total sum of the activation numbers for 6.6a is 24. This shows that 6.6b is less complex compared to 6.6a which predicts correctly the human performance. The relevant proof nets (figure 6.6) is redrawn with colorful annotation of the weights. Axioms with the weight 1 are bold in order to be distinguished easily from the axioms with the weight 0.

6.4.3 Quantifying Preferences with Satisfaction Ratio

We have focused on a narrow class of non-canonical utterances [PS01] which is incomplete sentences with some missing categories. Some studies [DPD09; Pro10; BP08; BP14], model the notion of the syntactic gradience. We can show this notion in the example 6.8 (taken from [DPD09]) which shows different sentences ordered by decreasing grammatical acceptability. Each given judgment corresponds to a percentage which estimates the acceptability compared with the reference sentence, namely 6.8a.

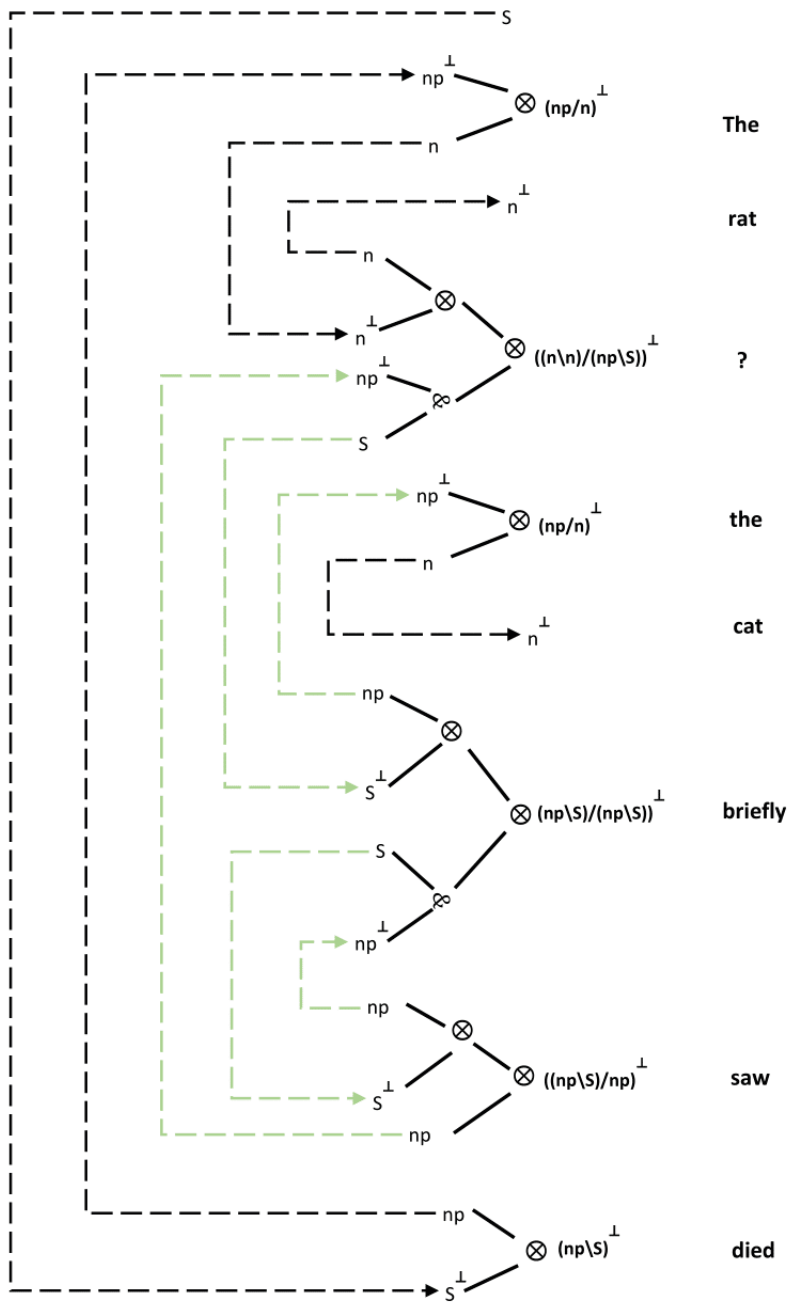


FIGURE 6.6: Proof net analysis for 6.7b with the weights.

Example 6.8.

6.8a. The employees have sent a very complete report to their employer. [100%]

6.8b. The employees have sent very complete report to their employer. [92.5%]

6.8c. The employees have sent a very complete report to. [67.5%]

6.8d. The employees a very complete report to their employer. [32.5%]

6.8e. The employees a very complete report. [5%]

The issue of modeling natural judgments of acceptability may be addressed by the five postulates (see references as they are reported in the [DPD09]) outlined below:

Postulate 1 (Failure Cumulativity): Gradience is impacted by by the number of constraints it violates.

Postulate 2 (Success Cumulativity): Gradience is impacted by the number of constraints it satisfies.

Postulate 3 (Constraint Weighting): Acceptability is impacted to a different extent according to which constraint is satisfied or violated.

Postulate 4 (Constructional complexity): Acceptability is impacted by the the complexity of the constituent structure.

Postulate 5 (Propagation): Acceptability is propagated through the dominance relationships; that is, an utterance's acceptability depends on its nested constituents' acceptability.

Before turning into the problem, we broadly provide a comparative review of the above postulates. An interesting question (aligned with the aims of this dissertation) is how one may apply those criteria on the categorial proof nets instead of other formalisms (such as Property Grammars[Bla00b; Bla00a]) which exist in the literature for modeling the grammaticality judgment. Let us start with the postulates (1)-(3). Since we are using categorial proof nets for our syntactic representation, the only way to use these criteria is to re-interpret the axioms link in the proof nets as constraints. This idea is not misleading at all. Axiom-links make a sort of constraints between the atomic categories in the words. But what about the violated/satisfied constraints? Our suggestion is to re-interpret the axiom links that are connected to the missing words as violated constraints and the axiom-links that are connected to available words as successful constraints. This assumption regardless of its predictive power (as we will see) can be supported by the fact that missing categories are a kind of violation in the syntactic representation while the existing order (by the assumption of in the problem we have stated) is in the grammatical order. Postulate (4) has been heavily studied in chapters 3 and 5. But, it is not suitable for the problem that now we are dealing with in this chapter. Postulate (5) is not investigated yet in this research, even though, capturing it to the categorial proof nets does not seem to be a straightforward case.

Definitions of the Satisfaction Ratio

Definition 6.18. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let $c - c'$ be an axiom in π such that $c \in C_i$ and $c' \in C_j$ $(i, j \in [1, n])$. We define the **violation weight** of the axiom $c - c'$ as 1 if $w_i : C_i$ or $w_j : C_j$ is a missing word which is suggested (by an external algorithm/source) to complete the sentence. Otherwise, the **violation weight** of the axiom $c - c'$ is 0.

Definition 6.19. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. The violation ratio of π, written as $f_{vr}(\pi)$, is the number of the all axiom-links minus the sum of the **violation weights** in π divided by the number of the axiom-links.

Definition 6.20. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is SR-preferred to π_j whenever $f_{sr}(\pi_i) > f_{sr}(\pi_j)$.

Example 6.9.

6.9a. The employees have sent a very complete report to their employer. [100%]

6.9b. The employees have sent very complete report to their employer. [77%]

6.9c. The employees have sent a very complete report to. [70%]

6.9d. The employees a very complete report to their employer. [64%]

6.9e. The employees a very complete report. [35%]

Now we can apply the Satisfaction ratio function to the examples 6.8a-6.8e rewritten here as 6.9a-6.9e with the new ratio in percentage. All the proof nets relevant to the examples are illustrated in the figures 6.7-6.11. As we can observe our definition correctly predicts the incremental syntactic acceptance that is gained in [DPD09]. Here are the detailed calculations for each example:

Satisfaction Ratio for 6.9a is $\frac{17-0}{17} = 100\%$

Satisfaction Ratio for 6.9b is $\frac{17-4}{17} = 77\%$

Satisfaction Ratio for 6.9c is $\frac{17-5}{17} = 70\%$

Satisfaction Ratio for 6.9d is $\frac{17-6}{17} = 64\%$

Satisfaction Ratio for 6.9e is $\frac{17-11}{17} = 35\%$

6.5 Conclusion and Future Works

We explained two algorithms: the first algorithm focused on the problem of finding one missing category in an incomplete utterance with $O(n^4)$ time complexity. We may have following extensions to the first algorithm:

The first extension is moving to non-rigid categories, namely fixed number of several categories assigned to each single word– instead of rigid ones. Since our approach benefits from chart parsing, this is, in principle, tractable. However, it will

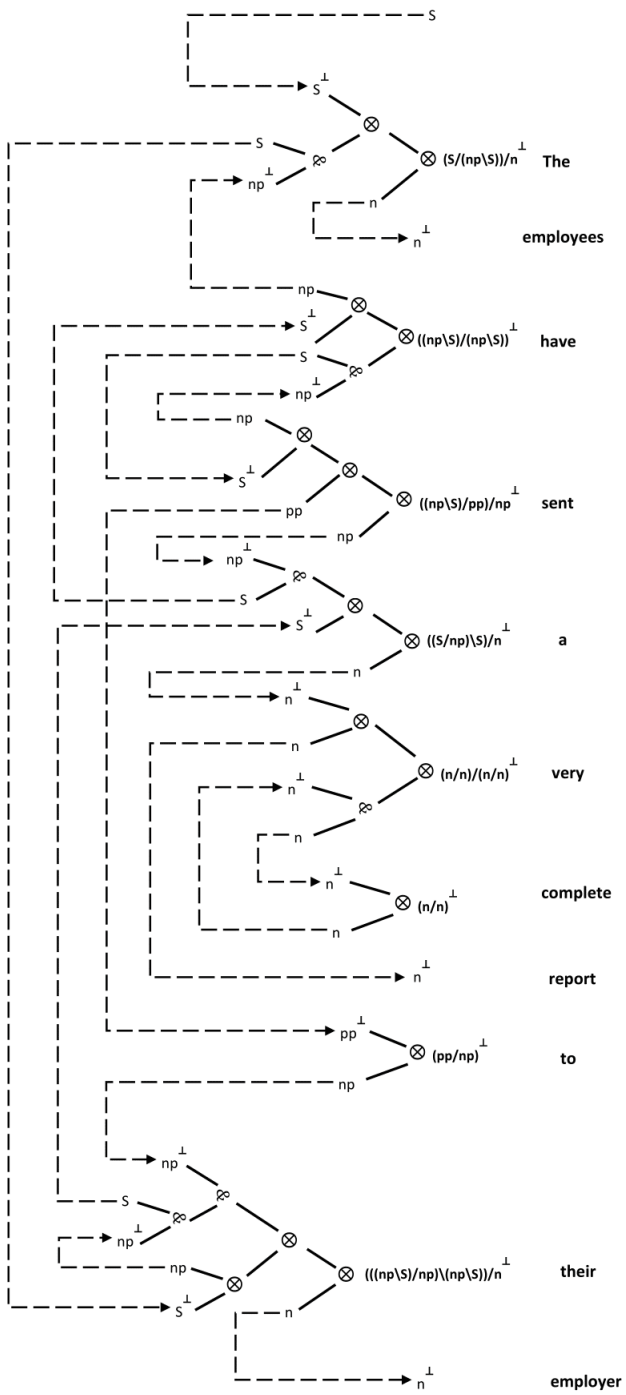


FIGURE 6.7: Proof net analysis for 6.9a with the (violation) weights.

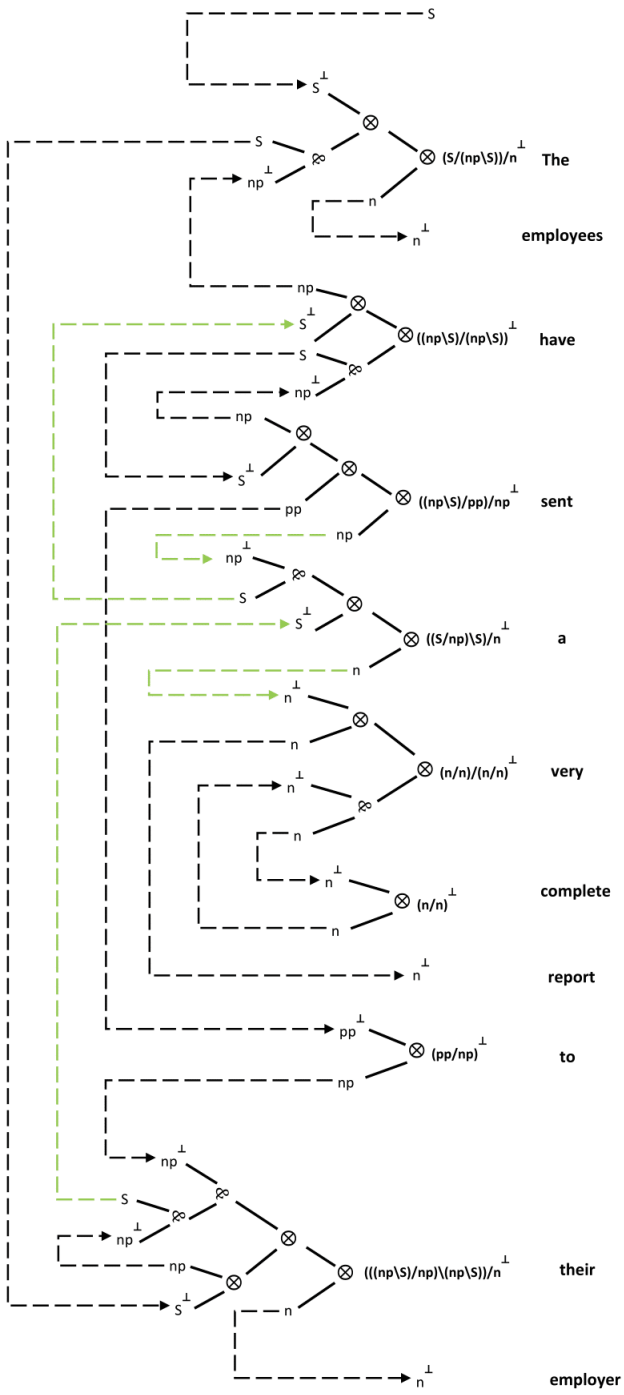


FIGURE 6.8: Proof net analysis for 6.9b with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0' .

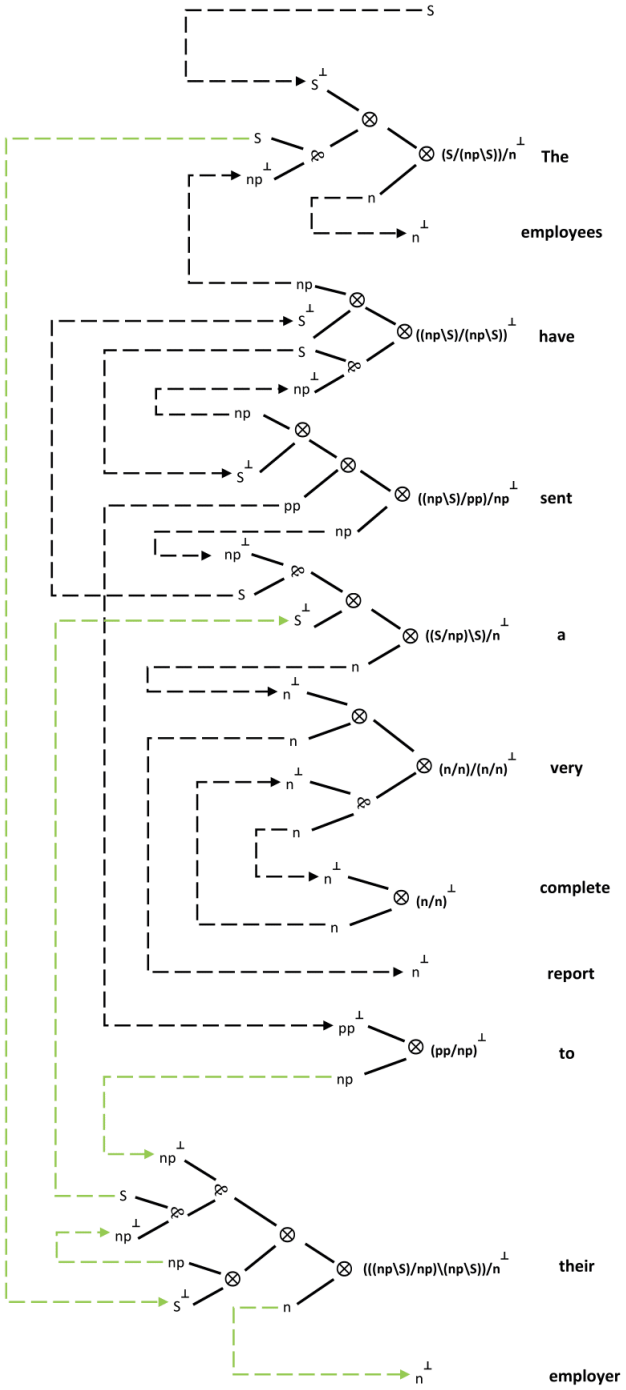


FIGURE 6.9: Proof net analysis for 6.9c with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0' .

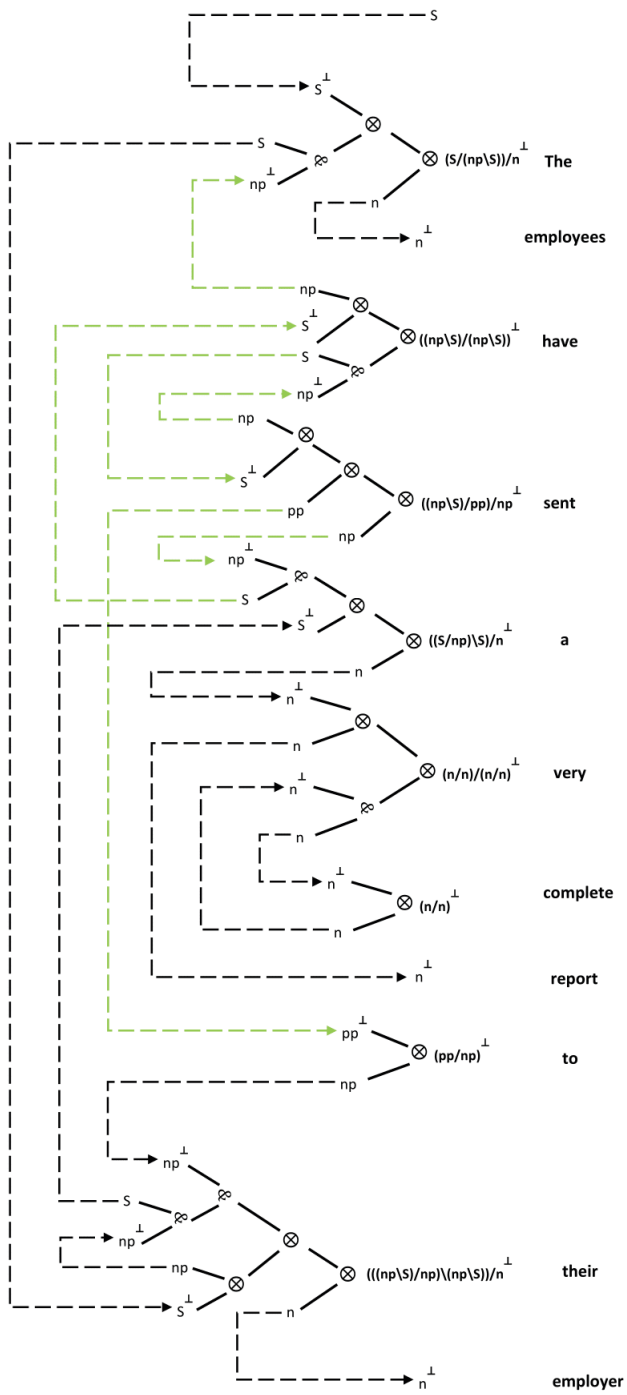


FIGURE 6.10: Proof net analysis for 6.9d with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0' .

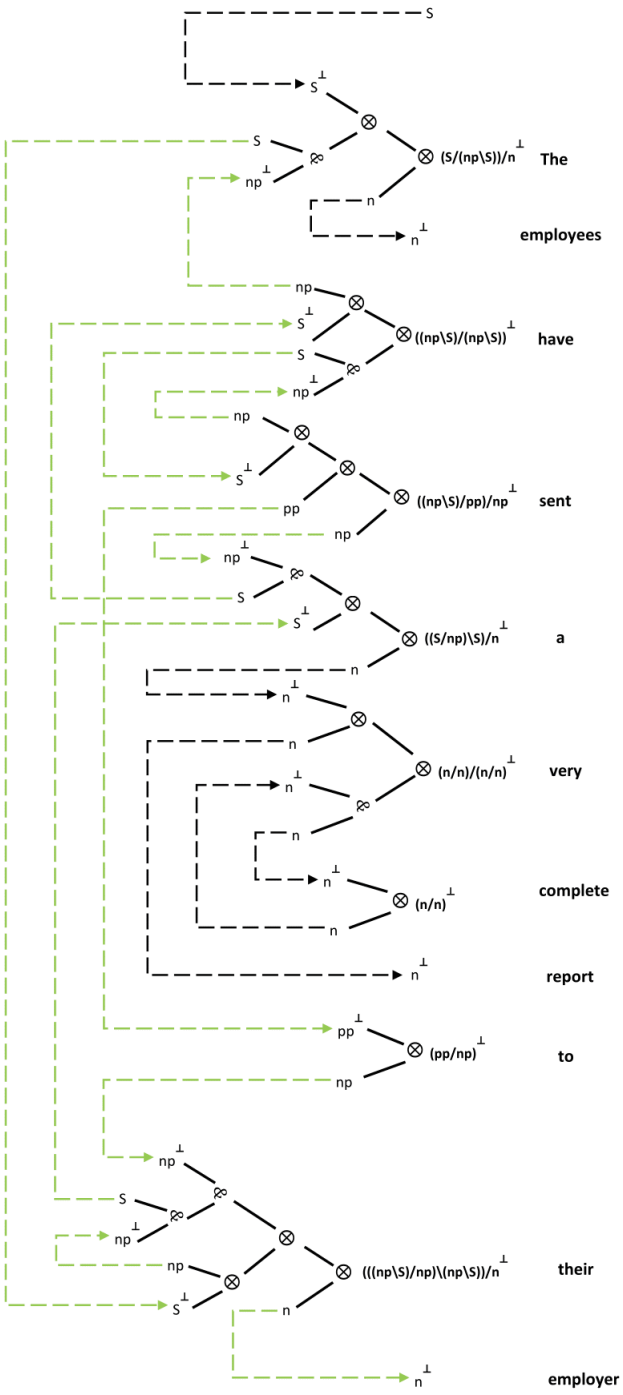


FIGURE 6.11: Proof net analysis for 6.9e with the (violation) weights. The violation weights with the light-colored axioms have the value '1'; the rest, i.e. the dark-colored axioms have the value '0'.

increase our computational complexity as it is expected. We can reduce our search space when we have lexical ambiguities (several categories for each word of the input sentence) by adopting super-tagging technique [BJ11]. The one which is very suitable for our purpose is Moot's super-tagger [Moo15] which is implemented in the wide-coverage version of the Grail parser.

The second extension is to focus on several numbers of missing categories, instead of just one. One of the possibilities would be using Jenson's approach [Jen+83] to build possible structural sub-trees of an incomplete utterance to find a limited number of suggestion for the position of missing categories. Also, for finding the exact missing words we can use some studies that bridge JDM with linguistic constraints [PL11]. Further study is required for more clarification.

The third extension is to tackle other classes of incomplete utterances such as extra categories, misused categories, swap categories or any possible combinations. The main problem with this approach is the need of having a computational mechanism to know what is the exact problem of a sentence before solving it. For instance, if we know that in a given utterance, we have the fixed numbers of missing words, extra words, and swap categories, then we can tackle the problem more efficiently with a different module that we might build.

The fourth extension is working on other families of categorial grammars such as Lambek's Calculus [Lam58] or some restricted version of Partial Proof Trees [JK97] for categorial grammar. The main problem –that the author is still dealing with– is that partial proof trees need to be restricted in order to work with Lambek's calculus. Moreover, designing a computationally decidable system with efficient complexity for incomplete utterances is still an on-going research. Generally speaking, this line of research can possibly be helpful in the domain of incoherence discourses. For instance, writing-assistant systems with questionnaires, for some specific targeted users such as patients suffering from specific mental disorders.

Our second algorithm benefits from Constraint Handling Rules and also the structured objects instead of strings, which has a lower generative capacity. The algorithm suggests generation trees as the solution of a given ungrammatical utterance. The outcome of our algorithm provides constructions that are lacking in other string-oriented error detecting approaches. CHRs can be easily implemented by available CHR features which exist widely in the mainstream programming languages. The algorithm introduced here can be extended and adapted to tackle other missing categories (for instance missing determiners and quantifiers); distinguishing and fixing misused words; acquiring more complex grammars. Interesting studies that are not yet properly integrated to our current approach [DB04b; DB05; AD16; DM12] can be beneficial for our future expansion.

Finally, we introduced two new metrics of the linguistic complexity for the different candidates that our algorithms can produce from a given incomplete sentence. Since we use categorial grammar as our syntactic analysis and categorial proof nets as our parsing, it was possible to introduce *Activation Theory* and *Satisfaction Ratio* for measuring the linguistic preferences in non-canonical utterances. This bridging between linguistic theories and the categorial proof nets would be integrated into our final model as we will discover in the next chapter.

Chapter 7

Putting It All Together: Preference over Linguistics Preferences

> You must understand there is more than one path to the top of the mountain.
>
> Miyamoto Musashi

7.1 Introduction

Our computational preferential modeling for natural language meanings has been explicated in the chapters 3-6. Now, we have to consider the problem of the integration of the suggested metrics and measurements in order to make them a unique model that can handle different problems instead of having different solutions of the different island of sub-problems. After all, this aggregation of the preferences can truly be considered as another problem of preference modeling. In a meta-theoretical sense, it is a meta-preference modeling over some available object-preference measurements by means of the categorial proof nets. In other words, we assume that the human choice is not just the result of preferences but it is a preference over other linguistic preferences.

The rest of the chapter is organized as follows: section 7.2 provides an overview of the preferential metrics that we have introduced in this dissertation. Section 7.3 introduces some procedures for integrating the linguistic difficulty metrics. In the section 7.4, we provide a general scheme that we need to make the final preference over different linguistic preferences. Section 7.5 provide some examples, to illustrate how our integrated procedure really works. We count the deficiencies of our proposal in the section 7.6, and in the last section, we conclude and we explain possible future works.

7.2 Complexity Metrics: Summaries

In this section, we summarize some of the computational preferential metrics that has been explicated in the chapters 3-6.

7.2.1 Quantifiers Order Measurement

We can gain quantifiers order measurement as follows:

1. Given the logical formula which corresponds to the left-to-right reading of the sentence; add an index from 1 to n to each quantifier from left to right obtaining $Q_1, Q_2 \ldots, Q_{n-1}, Q_n F$ call this formula Φ.

2. By the procedure introduced in [HS87, p.49-53], derive all the valid quantifier (scope) readings of the sentence.

3. Let $\xi_1, \ldots, \xi_m$ be rewritten formulas obtained from the previous step.[1]

4. Calculate for each ξ_i the penalty of quantifiers re-ordering as

$$f_{qr}(\xi_i) = \frac{1}{\sum_{j=1}^{n} \mid j - Pos(Q_j, \xi_i) \mid +1}$$

in which $Pos(Q_j, \xi_i)$ is the occurrence position of the quantifier Q_j in ξ_i counted from left to right and incremented from number one.

7.2.2 Incomplete Dependency Theory

We can rewrite IDT-based complexity profiling [Mor00] by the following definitions:

Definition 7.1. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i ($i \in [1, n]$). The incomplete dependency number of C_{i_0} in π, written as $ID_\pi(C_{i_0})$, is the count of axioms $c - c'$ in π such that $c \in (C_{i_0-m} \cup S)$ ($m \geq 0$) and $c' \in C_{i_0+n+1}$ ($n \geq 0$).

Definition 7.2. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. We define the IDT-based linguistic complexity of π, written $f_{idt}(\pi)$ by $f_{idt}(\pi) = (1 + \sum_{i=1}^{n} ID_\pi(C_i))^{-1}$.

Definition 7.3. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is IDT-preferred to π_j whenever $f_{idt}(\pi_i) > f_{idt}(\pi_j)$.

7.2.3 Dependency Locality Theory

Here are the relevant definitions of the DLT-based proposal on the basis of the categorial proof nets:

Definition 7.4. A word w is said to be a discourse referent whenever it is a *proper noun*, *common noun* or *verb*.

[1] As we discussed in 3.3.2, we can also gain the valid quantifier scope readings from semantic readings of the proof-nets (for Multiplicative Linear Logic). In this case we would have $\xi_1, \ldots, \xi_m$ which corresponds to their relevant syntactical proof-nets as $\pi_1, \ldots, \pi_m$. Since, Multiplicative Linear Logic provability is NP-complete, we have not proposed this solution as the proper treatment. Further investigation is required.

Definition 7.5. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let $c - c'$ be an axiom in π such that $c \in C_i$ and $c' \in C_j$ $(i, j \in [1, n])$. We define the **length** of axiom $c - c'$ as the integer $i + 1 - j$.

Definition 7.6. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i, and let consider axioms $c - c'$ with c in C_{i_0} and c' in some C_{i_0-k}. Let us consider the largest k for which such an axiom exists — this is the longest axiom starting from C_{i_0} with the previous definition. The dependency locality number of C_{i_0} in π, written as $DL_\pi(C_{i_0})$, is the number of discourse referent words between $w_{i_0} : C_{i_0}$ and $w_{i_0-k} : C_{i_0-k}$. The bound words, i.e. $w_{i_0} : C_{i_0}$ and $w_{i_0-k} : C_{i_0-k}$ should also be counted. Alternatively, it may be viewed as $k + 1$ minus the number of non-discourse references among those $k + 1$ words.

Definition 7.7. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. We define the DLT-based linguistic complexity of π, written $f_{dlt}(\pi)$ by $f_{dlt}(\pi) = (1 + \sum_{i=1}^{n} DL_\pi(C_i))^{-1}$.

Definition 7.8. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is DLT-preferred to π_j whenever $f_{dlt}(\pi_i) > f_{dlt}(\pi_j)$.

7.2.4 Activation Theory

Definition 7.9. A word w is said to be an **activator word** when it is a pre-modifier that activates the head phrase in an expression. The mentioned head is said to be the **activated word**.

Definition 7.10. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let $c - c'$ be an axiom in π such that $c \in C_i$ and $c' \in C_j$ $(i, j \in [1, n])$. We define the **activation weight** of the axiom $c - c'$ as 0 if $w_i : C_i$ or $w_j : C_j$ is an activator or activated word. Otherwise, the **activation weight** of the axiom $c - c'$ is 1.

Definition 7.11. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let C_{i_0} be one of the C_i $(i \in [1, n])$. The activation number of C_{i_0} in π, written as $AN_\pi(C_{i_0})$, is the sum of axioms **activation weights** $c - c'$ in π such that $c \in (C_{i_0-m} \cup S)$ $(m \geq 0)$ and $c' \in C_{i_0+n+1}$ $(n \geq 0)$.

Definition 7.12. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. We define the Activation-based linguistic complexity of π, written $f_{idat}(\pi)$ by $f_{idat}(\pi) = (1 + \sum_{i=1}^{n} AN_\pi(C_i))^{-1}$.

Definition 7.13. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is Activation-preferred to π_j whenever $f_{idat}(\pi_i) > f_{idat}(\pi_j)$.

7.2.5 Satisfaction Ratio

Definition 7.14. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. Let $c - c'$ be an axiom in π such that $c \in C_i$ and $c' \in C_j$ $(i, j \in [1, n])$. We define the **violation weight** of the axiom $c - c'$ as 1 if $w_i : C_i$ or $w_j : C_j$ is a missing word which is suggested (by an external algorithm/source) to complete the sentence. Otherwise, the **violation weight** of the axiom $c - c'$ is 0.

Definition 7.15. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$. The Satisfaction ratio of π, written as $f_{vr}(\pi)$, is the number of the all axiom-links minus the sum of the **violation weights** in π divided by the number of the axiom-links.

Definition 7.16. Given two syntactic analyses π_i and π_j, not necessarily of the same words and categories, we say that π_i is SR-preferred to π_j whenever $f_{sr}(\pi_i) > f_{sr}(\pi_j)$.

7.3 Integration of Linguistic Difficulty Metrics

We study some general strategies to integrate different linguistic complexity metrics that we have introduced up to now.

7.3.1 Integrating Dependency Locality and Incomplete Dependency Theories

Can DLT-based complexity profiling supersede the IDT-based complexity profiling? This question is crucial at this stage. The answer to this question is unfortunately *No*. We have to explain why, and also we should investigate new ways to overcome this problem. In the section 5 we showed that DLT-based complexity profiling can predict both referent-sensitive and some of the non-referent-sensitive phenomena. To count them, we can highlight *structures with embedded pronouns* as the referent-sensitive phenomenon, and *garden paths, unacceptability of center embedding, preference for lower attachment, and passive paraphrases acceptability* as the non-referent-sensitive phenomena. We will introduce one linguistic phenomenon that DLT-based complexity profiling can not support while IDT-based complexity profiling can do so, and it is *Heavy Noun Phrase Shift* [Mor00] as in the following example:

Example 7.1.

7.1a. John gave Bill the painting that Mary hated.

7.1b. John gave the painting that Mary hated to Bill.

The preference in these kinds of the sentences is not in different possible ambiguous readings, since the expressions are synonymous which holds semantics constant. The real preference is on the syntactic forms in which 7.1a is preferred to 7.1b. As

we can observe in the table 7.1, the IDT-based complexity for 7.1b is more complex than 7.1a while, in contrary, in the table 7.2, this is reversed. This shows the failure of DLT-based complexity profiling for *Heavy Noun Phrase Shift*.

7.1a	John 2	gave 2	Bill 1	the 1	painting 2	that 2	Mary 3	hated 0	
7.1b	John 2	gave 2	the 2	painting 3	that 3	Mary 4	hated 1	to 1	Bill 0

TABLE 7.1: Calculation of the incomplete dependency number for 7.1a and 7.1b.

7.1a	John 0	gave 2	Bill 0	the 0	painting 0	that 1	Mary 0	hated 2	
7.1b	John 0	gave 2	the 0	painting 0	that 1	Mary 0	hated 2	to 0	Bill 0

TABLE 7.2: Calculation of the dependency locality number for 7.1a and 7.1b.

The above problem can be overcome if we apply the following procedure for measuring a combined IDT/DLT-based complexity profiling:

Definition 7.17. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$ — that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$— such that the syntactic categories of the subjective pronouns, objective pronouns, reflexive pronouns and possessive pronoun are assigned as np. We can calculate the combined IDT/DLT-based complexity measurement of the π as the lexicographic orders $(f_{idt}(\pi), f_{dlt}(\pi))$.

We should notice that the type assignment are not for deriving the logical formula. It is for measuring the syntactic linguistic complexity measurement. The normal type shifted assignments are needed to get the logical formula. We will explore some examples in section 7.5 to investigate how this procedure really works on the real examples.

Definition 7.18. Let π_1 and π_2 be two syntactic analyses of s_1 and s_2, respectively which are not necessarily of the same words and categories. The s_1 is IDT/DLT-preferred over s_2 if $(f_{idt}(\pi_1), f_{dlt}(\pi_1)) > (f_{idt}(\pi_2), f_{dlt}(\pi_2))$.

7.3.2 Integrating Incomplete Dependency and Activation Theories

Can Activation-based complexity profiling supersede the IDT-based complexity profiling? The answer to this question is *Yes*. The definition of Activation-based complexity is designed in a way that IDT-based complexity profiling can be considered as a specific case of it. The proof is straightforward if we assign 1 as the activation weight to all the axiom links in a categorial proof nets, we can easily observe that it would be changed to the IDT-based complexity profiling measurement.

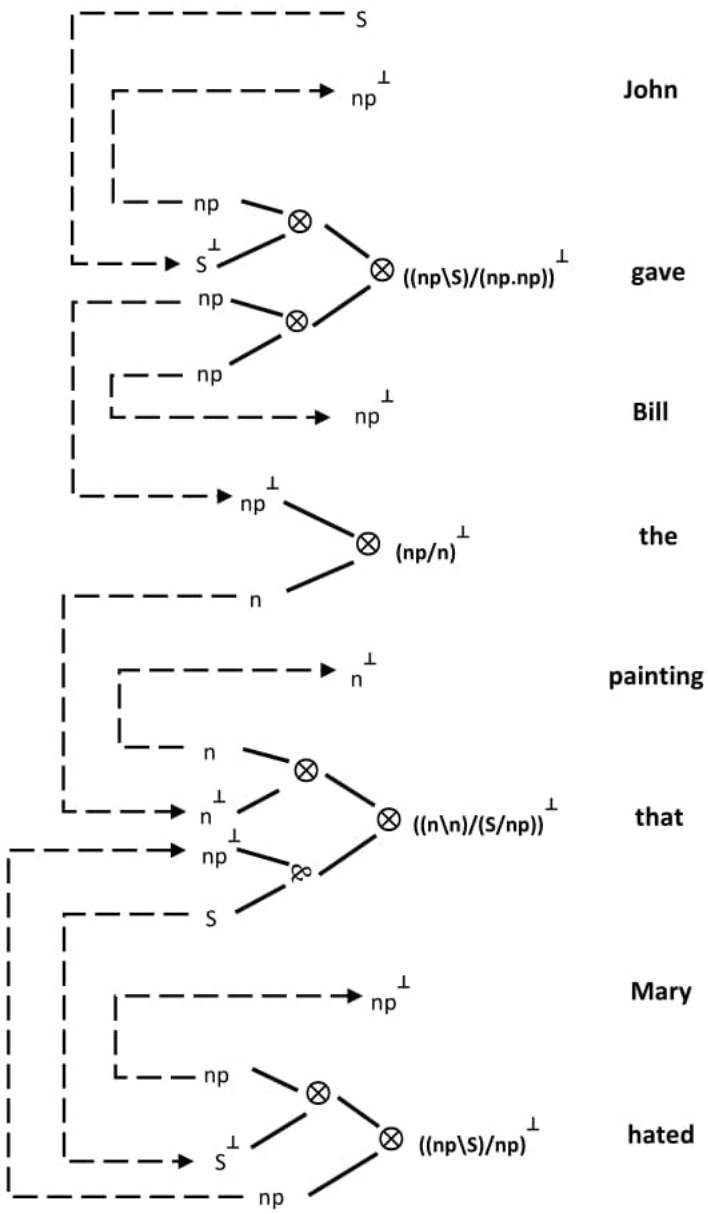


FIGURE 7.1: Proof net analysis for 7.1a.

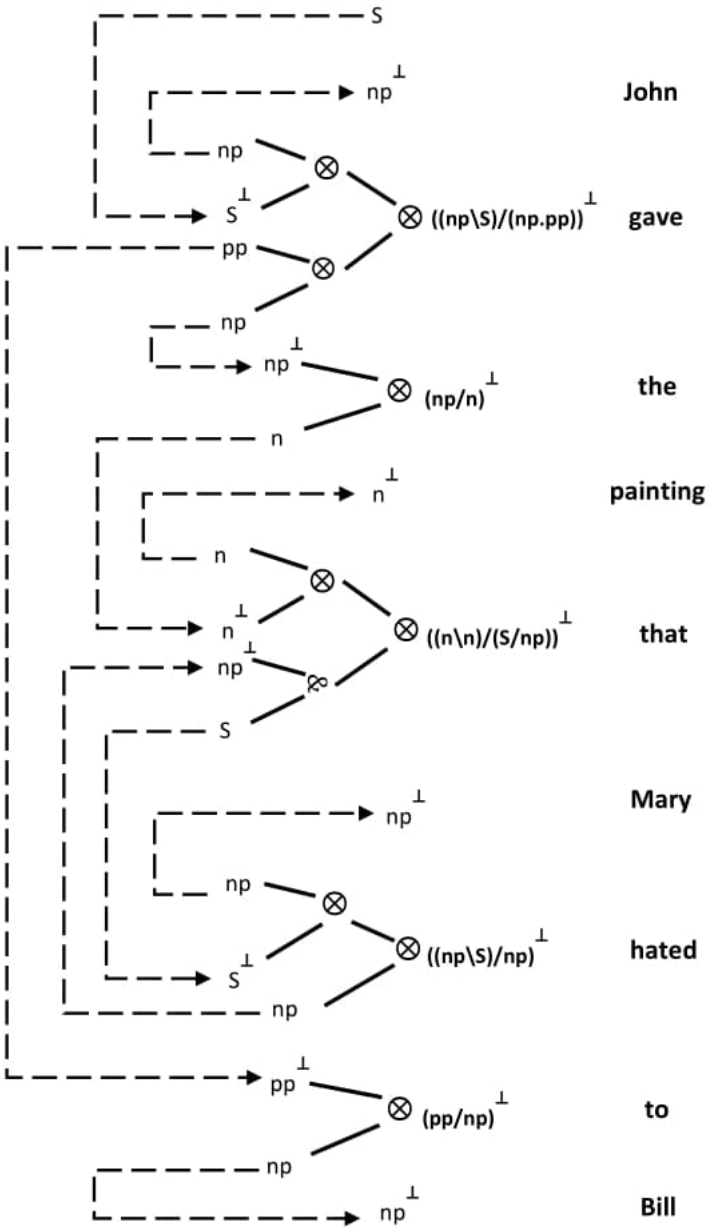


FIGURE 7.2: Proof net analysis for 7.1b.

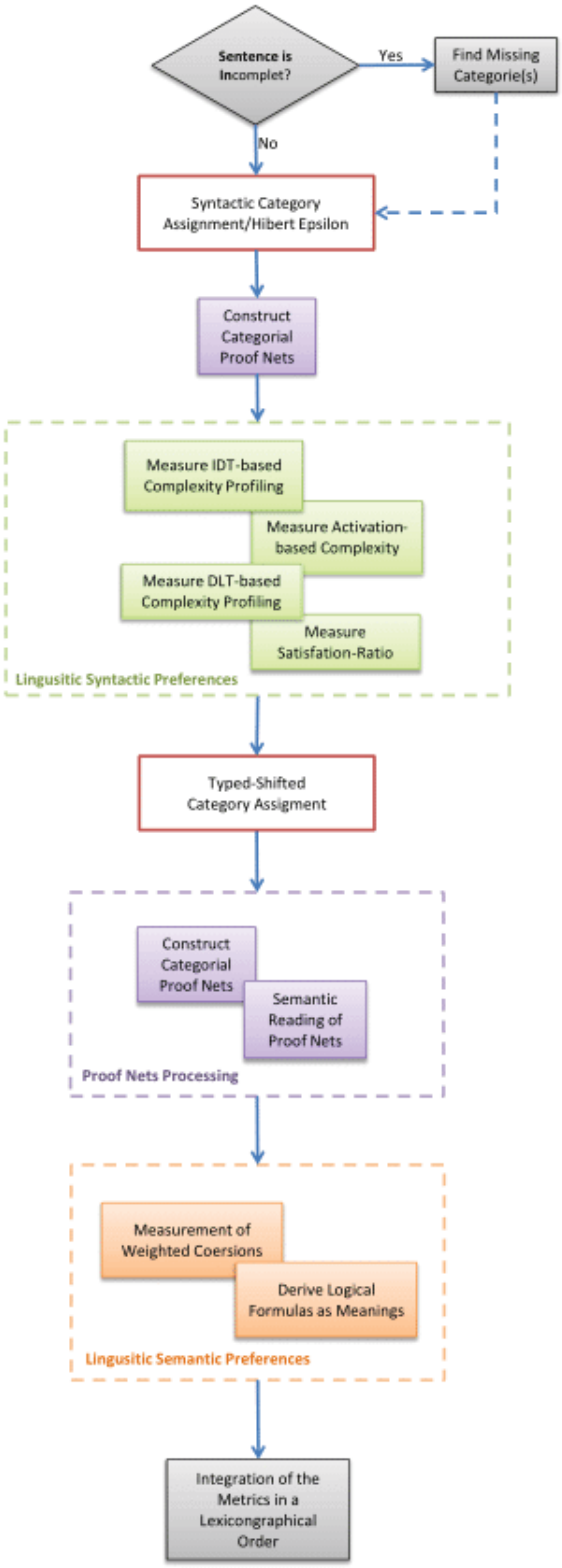


FIGURE 7.3: The General Scheme for Integrating the Rankings.

7.3.3 Integration of the Satisfaction Ratio

We do not need to design any integration procedure for the *Satisfaction Ratio*. This is due to this fact that Satisfaction Ratio is a specific task which is measuring some incomplete sentences with missing categories (interpreted as violation of grammatical rules) based on referencing a complete sentence.

7.4 Preference over Linguistic Preferences: A General Scheme

After understanding the philosophy behind the integration of difficulty measurements in the previous section, we can get back to the general scheme. Figure 7.3 shows the general scheme that we have to provide as a unique model for quantifying the final preference over the linguistic preferences introduced throughout the chapters. We also investigate our procedure by exploring some examples in the next section.

Definition 7.19. Let π be a a syntactic analysis of $w_1, \cdots, w_n$ with categories $C_1, \ldots, C_n$, that is a categorial proof net with conclusions $(C_n)^\perp, ..., (C_1)^\perp, S$, such that:

i) the syntactic categories of the subjective pronouns, objective pronouns, reflexive pronouns and possessive pronoun are assigned as np;

ii) the syntactic categories of the quantifiers, determiners and possessive adjectives are assigned as np/n;

and let $\Pi = \{\pi_1, \cdots, \pi_n\}$ be the set of all possible categorial proof-net analyses, after assigning proper type-shifted categories (such as determiners, quantifiers and subjective/objective pronouns,...), then we can have the following calculations:

1. Calculate the DLT-based complexity measurement of the π as $f_{dlt}(\pi)$.
2. Calculate the IDT/Activation-based complexity measurement of the π as $f_{idat}(\pi)$.
3. Calculate the IDT/Activation/DLT-based combined complexity measurement of the π as the lexicographic orders $(f_{idat}(\pi), f_{dlt}(\pi))$.
4. Calculate the Weighted Coercion by generating the set of compound orders for each coercion weight f_{c_i} [see the procedure in 4.5.2] as $((f_{idat}(\pi), f_{dlt}(\pi)), f_{c_i})$.[2]
5. Calculate the Quantifiers Re-order Measurement by generating the set of compound orders for each quantifier scope reading as $\xi_i \in [[\Pi]]$ as $(((f_{idat}(\pi), f_{dlt}(\pi)), f_{c_i}), f_{qr}(\xi_i))$.

Definition 7.20. Let π_1 and π_2 be two syntactic analyses of s_1 and s_2, respectively which are not necessarily of the same words and categories and fulfill the conditions (i) and (ii) of the previous definition. Let ξ_1 and ξ_2 be two quantifier scope readings, and f_{c_1} and f_{c_2} two coercion weights of s_1 and s_2, respectively. The s_1 is overall-preferred over s_2 if $(((f_{idat}(\pi_1), f_{dlt}(\pi_1)), f_{c_1}), f_{qr}(\xi_1)) > (((f_{idat}(\pi_2), f_{dlt}(\pi_2)), f_{c_2}), f_{qr}(\xi_2))$

[2] As we saw in 4.5.2, we are restricted to just one set of coercion wights for each sentence in this dissertation. Although, this simple assumption works in this practice, it is simply far from real situation in the natural language. This shortcoming is considered as a future study case and deserve further investigation.

7.5 Some Examples

In this section, we examine the procedures 7.19 and 7.20 over different examples introduced up to now. Our target is to show that the unified model can preserve the same preferences that are linguistically expected. We repeat the examples for convenience, but, we just refer to the available calculations throughout chapters.

We start with the example 7.1a and 7.1b repeated here as 7.2a and 7.2b, respectively.

Example 7.2.

7.2a. John gave Bill the painting that Mary hated.

7.2b. John gave the painting that Mary hated to Bill.

Figures 7.1 and 7.2 show the syntactic categorial proof nets for the examples 7.2a and 7.2b which we label as π_1 and π_2, respectively. As we can observe $(f_{idat}(\pi_1), f_{dlt}(\pi_1)) = (0.07, 0.17)$ and $(f_{idat}(\pi_2), f_{dlt}(\pi_2)) = (0.05, 0.17)$. Thus, $(f_{idat}(\pi_1), f_{dlt}(\pi_1)) > (f_{idat}(\pi_2), f_{dlt}(\pi_2))$ which correctly predicts the syntactic preference of 7.2a over 7.2b. Notice that we have no quantifier and determiner, and this would not change the overall preference.

The next example is 3.4 repeated here as 7.3.

Example 7.3. Every researcher of a company saw a sample.

7.3a. $\forall x(Res(x) \wedge \exists y(Com(y) \wedge of(x,y)) \rightarrow \exists z(Sam(z) \wedge Saw(x,z)))$

7.3b. $\exists z(Sam(z) \wedge \forall x(Res(x) \wedge \exists y(Com(y) \wedge of(x,y) \rightarrow Saw(x,z)))$

7.3c. $\exists y(Com(y) \wedge \forall x(Res(x) \wedge of(x,y) \rightarrow \exists z(Sam(z) \wedge Saw(x,z)))$

7.3d. $\exists y(Com(y) \wedge \exists z(Sam(z) \wedge \forall x(Res(x) \wedge of(x,y)) \rightarrow Saw(x,z)))$

7.3e. $\exists z(Sam(z) \wedge \exists y(Com(y) \wedge \forall x(Res(x) \wedge of(x,y)) \rightarrow Saw(x,z)))$

7.3f. ?? $\forall x(Res(x) \wedge \exists z(Sam(z) \wedge of(x,y)) \rightarrow \exists y(Com(y) \wedge Saw(x,z)))$

Figures 3.2-3.6 show the syntactic categorial proof nets for the meaning representation of the examples 7.3a-7.3e which we label as π_1-π_5, respectively. Assigning the np/n to the quantifiers and determiners would give us the same syntactic analyses for all the readings as below:

$$(f_{idat}(\pi_1), f_{dlt}(\pi_1)) = (0.05, 0.17)$$

$$(f_{idat}(\pi_2), f_{dlt}(\pi_2)) = (0.05, 0.17)$$

$$(f_{idat}(\pi_3), f_{dlt}(\pi_3)) = (0.05, 0.17)$$

$$(f_{idat}(\pi_4), f_{dlt}(\pi_4)) = (0.05, 0.17)$$

$$(f_{idat}(\pi_5), f_{dlt}(\pi_5)) = (0.05, 0.17)$$

After calculation of the quantifier ordering cost, we would have:

$$((f_{idat}(\pi_1), f_{dlt}(\pi_1), f_{qr}(\pi_1)) = ((0.05, 0.17), 1)$$

$$((f_{idat}(\pi_2), f_{dlt}(\pi_2), f_{qr}(\pi_2)) = ((0.05, 0.17), 0.2)$$

$$((f_{idat}(\pi_3), f_{dlt}(\pi_3), f_{qr}(\pi_3)) = ((0.05, 0.17), 0.33)$$

$$((f_{idat}(\pi_4), f_{dlt}(\pi_4), f_{qr}(\pi_4)) = ((0.05, 0.17), 0.2)$$

$$((f_{idat}(\pi_5), f_{dlt}(\pi_5), f_{qr}(\pi_5)) = ((0.05, 0.17), 0.2)$$

The above complex lexicographical orders predict exactly what we discussed in the section 3. Notice that the effect of weighted coercions that we do not have it in this example is constant and it would not change the final result of the preference. The next example is 3.6 repeated here as 7.4.

Example 7.4. Everyone repairs something expertly.

7.4a. $exp(\forall x \exists y Repair(x, y))$

7.4b. $exp(\exists y \forall x Repair(x, y))$

Figures 3.7 shows two syntactic categorial proof nets for the meaning representation of the examples 7.4a and 7.4b which we label as π_1 and π_2, respectively. Assigning the np/n to the quantifiers and determiners would give us the same syntactic analyses for all the readings as below:

$$(f_{idat}(\pi_1), f_{dlt}(\pi_1)) = (0.13, 0.33),$$

$$(f_{idat}(\pi_2), f_{dlt}(\pi_2)) = (0.13, 0.33)$$

After calculation of the quantifier ordering cost, we would have:

$$((f_{idat}(\pi_1), f_{dlt}(\pi_1), f_{qr}(\pi_1)) = ((0.13, 0.33), 1)$$

$$((f_{idat}(\pi_2), f_{dlt}(\pi_2), f_{qr}(\pi_2)) = ((0.13, 0.33), 0.33)$$

The above complex lexicographical orders correctly predict the preference 7.4a over 7.4a. The next examples are 4.2a and 4.2b repeated here as 7.5a and 7.5b, respectively.

Example 7.5.

7.5a. A woman finished Villani's book.

7.5b. ?A goat finished Villani's book.

Assigning the np/n to the determiner would give us the same syntactic analyses for the examples 7.5a and 7.5b which we label as π_1 and π_2, respectively:

$$(f_{idat}(\pi_1), f_{dlt}(\pi_1)) = (0.10, 0.33)$$

$$(f_{idat}(\pi_2), f_{dlt}(\pi_2)) = (0.10, 0.33)$$

The calculation of the quantifier effect would not make any preference as it can be observed in the final orders:

$$f_{qr}(\pi_1) = f_{qr}(\pi_2) = 1$$

The calculation of the weighted coercion (see detailed discussion in 4.5.2) would finally provide the final preferences as below:

$$(((f_{idat}(\pi_1), f_{dlt}(\pi_1)), f_c^{reading}), f_{qr}(\pi_1)) = (((0.10, 0.33), 680), 1)$$

$$(((f_{idat}(\pi_1), f_{dlt}(\pi_1)), f_c^{writing}), f_{qr}(\pi_1)) = (((0.10, 0.33), 210), 1)$$

$$(((f_{idat}(\pi_2), f_{dlt}(\pi_2)), f_c^{eating}), f_{qr}(\pi_2)) = (((0.10, 0.33), 30), 1)$$

As we can observe the reading *A woman finished (reading) Villani's book* is preferred over *A woman finished (writing) Villani's book*. Also, the reading *A goat finished (eating) Villani's book* has less confident weight comparing to the first two readings, and this is exactly what we expect (see detailed discussion in 4.5.2).

The next examples are 5.1b and 5.1c repeated here as 7.6a and 7.6b, respectively.

Example 7.6.

7.6a. The reporter [who the senator [who John met] attacked] disliked the editor].

7.6b. The reporter [who the senator [who I met] attacked] disliked the editor].

Figures 5.5 and 5.7 show the syntactic categorial proof nets for the examples 7.6a and 7.6b which we label as π_1 and π_2, respectively. As we can observe $(f_{idat}(\pi_1), f_{dlt}(\pi_1)) = (0.02, 0.07)$ and $(f_{idat}(\pi_2), f_{dlt}(\pi_2)) = (0.02, 0.08)$. Thus, $(f_{idat}(\pi_2), f_{dlt}(\pi_2)) > (f_{idat}(\pi_1), f_{dlt}(\pi_1))$ which correctly predicts the syntactic preference of 7.6b over 7.6a. Notice that we have an equal constant quantifier and determiner effect, and this would not change the overall preference.

The last examples are 6.7a and 6.7b repeated here as 7.7a and 7.7b, respectively.

Example 7.7.

7.7a. *The rat the cat saw died.

7.7b. *The rat the cat briefly saw died.

Figures 6.4 and 6.6 show the syntactic categorial proof nets for the examples 7.7a and 7.7b which we label as π_1 and π_2, respectively. As we can observe $(f_{idat}(\pi_1), f_{dlt}(\pi_1)) = (0.04, 0.11)$ and $(f_{idat}(\pi_2), f_{dlt}(\pi_2)) = (0.05, 0.11)$. Thus, $(f_{idat}(\pi_2), f_{dlt}(\pi_2)) > (f_{idat}(\pi_1), f_{dlt}(\pi_1))$ which correctly predicts the syntactic preference of 7.6b over 7.6a. Notice that we have an equal constant quantifier and determiner effect, and this would not change the overall preference.

7.6 Limitations

There are two limitations with respect to our proposal for aggregating the linguistic preferences. The first limitation is that our integrated model assumes that some factors are constant when we measure a specific linguistic phenomenon—as we saw in the examples of this chapter. Even if we change all the factors at the same time (with some kind of modeling) we would lack some linguistic/psycholinguistic theories to support our proposal. Our simple strategy of using compound lexicographical order seems good for modeling the psycholinguistic theories introduced up to now, but, it is not powerful enough for real-life complicated situations.

The second limitation is more general than the first one and it is the lack of a data-set for human language preferences. Ideally, one would have a data-set annotated for human-preferred readings of various naturally occurring utterances and then evaluate the proposed method to see how it would work over that data-set. This seems as a separate project that would be more interesting if it could be merged with

the natural language inference data-sets [Cha+17b]. For instance to evaluate those natural language inferences that are dependent on the meaning preferences. This is investigated in some studies and seems like an important task in natural language processing and computational linguistics[DGrt].

7.7 Conclusion

We discussed the problem of aggregation of the linguistic preferences. We tried to capture this notion by a procedure that uses the lexicographical orders of the linguistic/psycho-linguistic metrics. We first studied how the integration can be possible for the pairs of metrics and we eventually built the mentioned procedure. In the previous section, we suggested building a data-set for human language preferences and we discussed very generally how this idea would help to evaluate the natural language inferences that are dependent on the meaning preferences. This obviously goes beyond this research and demands new line of surveys as our future studies.

Chapter 8

Conclusion

> A man cannot understand the art he is studying if he only looks for the end result without taking the time to delve deeply into the reasoning of the study.
>
> Miyamoto Musashi

In this work, we have addressed (part of) the problem concerned with a human-like ranking of the meaning interpretations of ambiguous natural language utterances. The type of the sentences that we worked on includes canonical sentences and also the sentences with only missing categories as one potential type of the non-canonical utterances. In order to fulfill this goal, we suggested to use/extend Categorial Grammars for our syntactical analysis and Categorial Proof Nets as our syntactic parse. We also used Montagovian Generative Lexicon for deriving multi-sorted logical formula as our semantical meaning representation. This simple strategy let us apply different linguistic/psycholinguistic metrics quite easily in the syntactic, lexical and semantic layers—as we did by introducing different procedures/algorithms.

Our contributions are five-folded:

- The first contribution is regarding ranking the multiple-quantifier scoping problem. We introduced a new metric for quantifier reordering cost which is inspired by underspecified Hilbert's epsilon operator.
- The second contribution is on modeling the semantic gradience in sentences that have implicit coercions in their meanings. In order to perform such a task, we suggested to use a framework called Montagovian Generative Lexicon. We introduced a lexicon building procedure that incorporates types and coercions using crowd-sourced lexical data gathered by a serious game called JeuxDeMots.
- The third contribution provides a new locality-based referent-sensitive metric for measuring linguistic complexity by means of Categorial Proof Nets. The proposed metric correctly predicts some performance phenomena such as structures with embedded pronouns, garden pathing, unacceptability of center embedding, preference for lower attachment and passive paraphrases acceptability.
- The fourth contribution focus on incomplete utterances with missing categories. In the first place, we introduce two algorithms for resolving incomplete utterances. The first algorithm is based on AB grammars, Chart-based Dynamic Programming and learning Rigid AB grammars while the second

one employs Constraint Handling Rules which deals with $k > 1$ missing categories. Then, we introduced measurements on the fixed categorial proof nets motivated by Gibson's distance-based theories, Violation Ratio factor and Activation Theory.

- The last contribution focus on the problem of integrating different computational metrics for ranking preferences with a unique model. We see this problem similar, to some extent, to the problem of aggregation of the preferences.

8.1 Summaries

Figure 8.1 shows the problems and the relevant solutions that we have investigated in this dissertation. As it is illustrated, there are generally six different problems that we tried to solve in different chapters. What follows is a quick summary of the problems:

1. For the problem of the ranking quantifier scopes we suggested the procedure 3.4.2 in the chapter 3.
2. For the problem of the syntactic gradience we suggested the procedure 6.4.3 in the chapter 6.
3. For the problem of the semantic gradience we suggested the procedure 4.4 in the chapter 4.
4. For the problem of the linguistic complexity we suggested the definitions 5.4 and 6.4 in the chapters 5 and 6.
5. For the problem of the sentence completion we suggested two algorithms 6.2 and 6.3 in the chapter 6.
6. For the problem of the reference over linguistic preferences we suggested the procedure 7.4 in the chapter 7.

8.2 Future Works

We have following extensions and future works that seems to be interesting research directions:

8.2.1 Ontological-based Modeling of the Quantifier Scoping Preference

As we explicated in the chapter 3, our modeling on the quantifiers order— which is based on the surface syntax —does not always give us the exact human preference interpretation, and it may be superseded by the lexical or world knowledge. As in the example 8.1, the logical formula represented in the 8.1a is not the meaning preferred by the human, although, the existential quantifier has the wider scope. The preferred meaning is 8.1b, since our common sense knowledge tells us that each door-step of a museum needs a unique guard. This is a linguistic phenomenon that is really hard to be computationally modeled.

Example 8.1. A guard stands in front of each museum door-step.

8.1a. $\exists x(Guard(x) \wedge \forall y(Museum_Door_Step(y) \rightarrow Stand_In_Front_Of(x,z))$

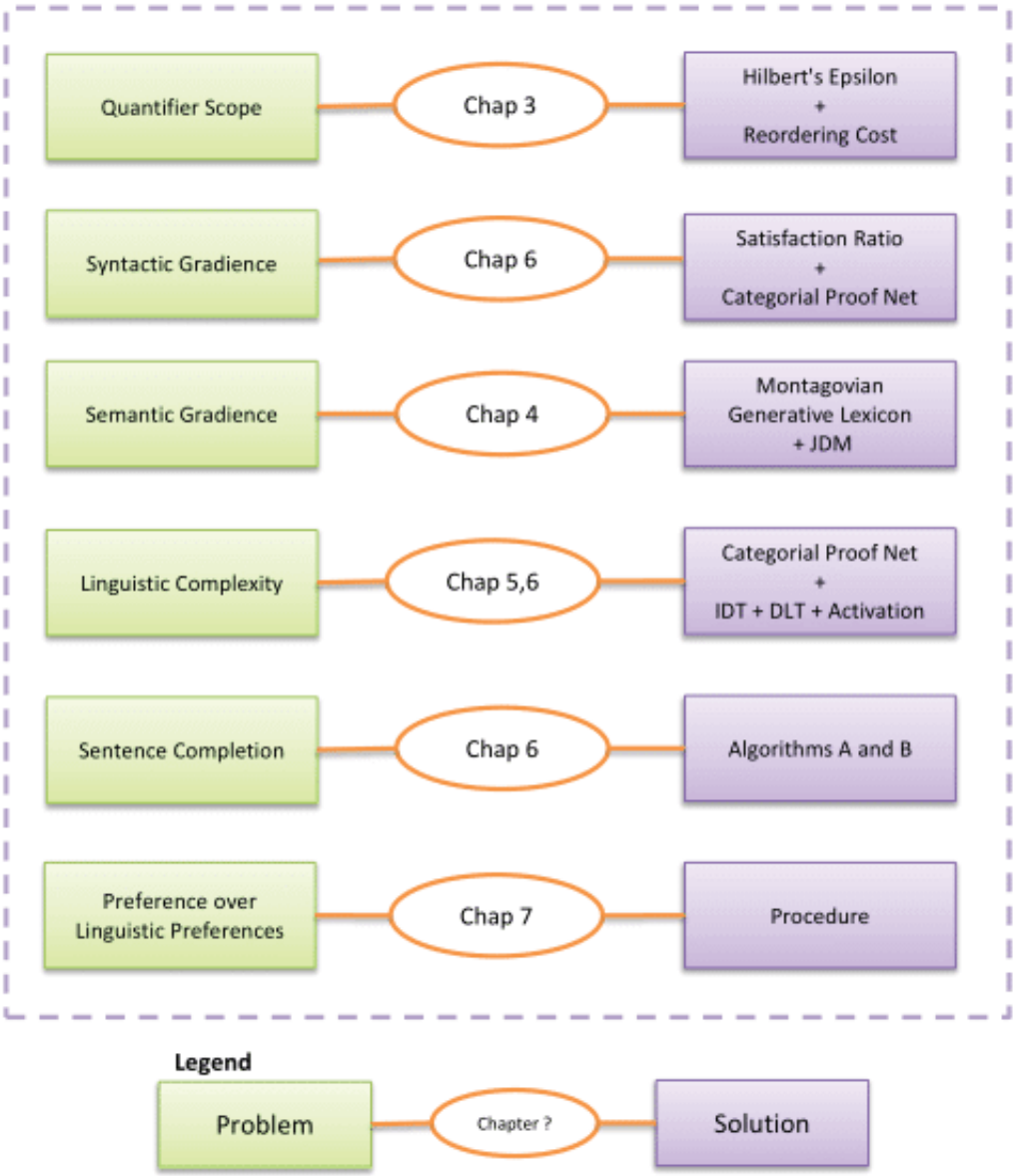


FIGURE 8.1: Summary of the problems and the relevant solutions.

8.1b. $\forall y(Museum_Door_Step(y) \rightarrow \exists x(Guard(x) \wedge Stand_In_Front_Of(x,z)))$

To overcome this problem we suggest to use the ontology-based interpretation of natural language[CUM14]. This would let us preserve good logical properties of our representation while permits us to have real-world knowledge to be integrated to our knowledge-based reasoning.

8.2.2 Annotated Data-set for Language Meaning Preferences

We discussed a serious limitation in the chapters 3 and 7 which exists in our rather young field of research, i.e. modeling/generating weighted logical formula. It is the lack of a data-set for human language preferences. Ideally, one would have a data-set annotated for human-preferred readings of various naturally occurring utterances and then evaluate the proposed method to see how it would work over that data-set. This seems as a separate project that would be more interesting if it could be merged with the natural language inference data-sets [Cha+17b]. For instance to evaluate those natural language inferences that are dependent on the meaning preferences. This is investigated in some studies [Pat16; KPP14] and seems like an important task in natural language processing and computational linguistics[DGrt].

8.2.3 Integrating Multiple Coercions in Montagovian Generative Lexicon

We discussed briefly in the chapter 4, a complete step-by-step procedure for analyzing the multi-sorted logical formulas from the sentence and its relation with the linguistic notion of semantic gradience. What remained as a future work is the integration of the various components by modifying the existing Montagovian Generative Lexicon framework for allowing multi-part coercions to be added, as a composition of transformations licensed from different lexemes. This should be examined in detail, as well as the implications of this modification to the time complexity of the computation, and the expressive power of the resulting formalism.

8.2.4 New Metrics for Linguistic Meaning Complexity

We introduced new metrics on syntactic linguistic complexity. This happened mainly by means of categorial proof nets. Nevertheless, we have not done any kind of studies on the complexity of the logical formulas as the object of meaning complexity at all. For instance, we can take the logical structure, i.e., par-links, tensor-links and the correctness criterion. This is important indeed because this structure is needed to compute the logical form (semantics) from the syntactic structure given by proof nets. For instance, nesting Lambek slashes (that are linear implications, and therefore par-links in the proof net) corresponds to higher order semantic constructions (e.g. predicates of predicates) and consequently this nesting of par-links increases the complexity of the syntactic and semantic human processing. We can also get inspired by other researches [Szy16] that uses other structures instead of the categorial proof nets. This study explores the semantic complexity of natural language and extends semantic theory with computational and cognitive aspects by combining linguistics and logic with computations and cognition.

8.2.5 Enhancing Sentence Completion Algorithms

We introduced two algorithms in the chapter 6. There are some techniques that might be useful for extension of the algorithms. We can reduce our search space when we have lexical ambiguities (several categories for each word of the input

sentence) by adopting super-tagging technique [BJ11]. The one which is very suitable for our purpose is Moot's super-tagger [Moo15] which is implemented in the wide-coverage version of the Grail parser. The other extension is to focus on several numbers of missing categories, instead of just one. One of the possible approaches would be using Jenson's approach [Jen+83] to build possible structural sub-trees of an incomplete utterance to find a limited number of suggestion for the position of missing categories. Another extension is to tackle other classes of incomplete utterances such as extra categories, misused categories, swap categories or any possible combinations.

Appendix A

Mathematical Proofs of the Properties of Algorithm B

This appendix provides the detailed mathematical proofs of the properties of algorithm B which is described in the chapter 6.

A.1 Proof of the Property (A)

Property (A): For every h, for every partial derivation tree t with height h, yield w and root r of grammar G, such that, each leave label is of NT, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

Proof:
We define $P(h)$:= *for every partial derivation tree t with height h, yield w and root r of grammar G, such that, each leave label is of NT, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$*. We prove (A) by strong induction on h. Recall that the grammar is assumed to be in CNF.

I. Base Case: It should be shown that P(0) holds, i.e. for every partial derivation tree t of height zero, yield w and root r, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$. Since, the derivation tree has height zero, so, there is only one leaf-node in the tree ,namely, r, so an empty sequence of CHR rules yields $(1^r, 0, \ldots, 0)$, and this proves P(0). Therefore, the base case holds.

II. Induction step: Assuming $P(i)$ is true for all $i \leq K$, it should be shown that $P(K+1)$ is true:

Let t be a partial derivation (binary) tree with height $K+1$, yield w and root r. Let t_1 and t_2 be two subtrees of t connecting to left and right sides of r such that r_1 and r_2 (=roots of t_1 and t_2 respectively) are children of r. So, based on the definition we have:

$$height(t) = max(height(t_1), height(t_2)) + 1$$

So, we can conclude:

$$max(height(t_1), height(t_2)) = K$$

This shows that t has two partial derivation subtrees where one of them is of height K and the other of height $K' \leq K$. Since both the K and K' are less or equal to K, based on the assumption of induction step, we can conclude that for t_1 and t_2 there exists sequences of CHR rules that yields $(1^{r_1}, 0, \ldots, 0)$ and $(1^{r_2}, 0, \ldots, 0)$ respectively.(*)

Let p be a production rule such that L.H.S. is r and R.H.S are the children of r in t, since all of the productions are assumed to be in CNF, we have four possible forms for p as follow: (Assuming X,Y and Z are distinct):

(i) The form of p is $r \rightarrow r_1 r_2$: Based on the definition, we can turn this rule into a CHR rule that decreases the number of r_1 and r_2 by one, and increases the number of r by one, so, the final status applying the CHR rules would be $(1^r, 0, \ldots, 0)$.

(ii) The form of p is $r \rightarrow r_1 r_1$: Based on the definition, we can turn this rule into a CHR rule that decreases the number of r_1 by two, and increases the number of r, so, the final status applying the CHR rules would be $(1^r, 0, \ldots, 0)$.

(iii) The form of p is $r \rightarrow r r_1 (or\ r \rightarrow r_1 r)$: Based on the definition, we can turn this rule into a CHR rule that decreases the number of r_1 by one, performs nothing to the number of r, so, the final status applying the CHR rules would be $(1^r, 0, \ldots, 0)$.

(iv) The form of p is $r \rightarrow rr$: Based on the definition, we can turn this rule into a CHR rule that only decreases the number of X by one, so, the final status applying the CHR rules would be $(1^r, 0, \ldots, 0)$.

The above analyses of all possible productions shows the existence of CHR rules that ends up to $(1^r, 0, \ldots, 0)$. (**)

Having (*) and (**) we can conclude that for every partial derivation tree t of height $K+1$, yield w and root r, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$. So Induction step, i.e., $\forall i \leq K\ P(i) \Rightarrow P(K+1)$ holds.

III. Conclusion: Based on (I) and (II), and the principle of strong induction we can conclude that $\forall h \geq 0\, P(h)$ holds, i.e., for every h, for every partial derivation tree t with height h, yield w and root r of grammar G, such that, each leave label is of NT, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

A.2 Proof of the Property (B)

Property (B): For every partial derivation tree t with yield w and root r of grammar G, such that, each leave label is of NT, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

Proof:
If we assume (A) and (A)⇒(B) then, the proof of (B) would be straightforward: By one application of modus ponens we can conclude (B). But, firstly, we need to prove the very assumptions that we have used in the general scheme.

Proof of (A) ⇒ (B)
Let t_1 be a derivation tree, let h_1 be the height of t_1, yield w and root r such that each leave label is from NT, because of (A) there exist a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$, therefore, statement(B) holds, and it can be concluded that (A) ⇒ (B).

A.3 Proof of the Property (C)

Property (C): For every sequence w of NT^* produced by grammar G from r, there exists a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$.

Proof:
If we assume (B) and (B)⇒(C); then, the proof of (C) would be straightforward: By one application of modus ponens we can conclude (C). But, firstly, we need to prove the very assumptions that we have used in the general scheme.

Proof of (B) ⇒ (C)
Let w be a sequence of NT^* produced by G, there is a derivation tree t with root r for w. Since, the yield of t is w then each leave label is from NT. Based on (B) there exist a sequence of CHR rules that yields $(1^r, 0, \ldots, 0)$ starting from $\vec{w}$, therefore, (C) holds, and it can be concluded that (B) ⇒ (C).

A.4 Proof of the Property (D)

Property (D): (A) If a multiplicity vector $\vec{t_i}$ converts to a vector $\vec{t_{i+1}}$ under only the descending rules of a given CHR(G), then, $d(\vec{t_{i+1}}) < d(\vec{t_i})$.

(B) If a multiplicity vector $\vec{t_i}$ converts to a vector $\vec{t_{i+1}}$ under only the ascending rules of a given CHR(G), then, $d(\vec{t_{i+1}}) \leq d(\vec{t_i})$.

Proof:
(A) If a multiplicity vector $\vec{t_i}$ converts to a vector $\vec{t_{i+1}}$ under only the descending rules of a given CHR(G), then, there are two possibilities for CFG rules (introduced in section 3.2):

(i) $A_i \rightarrow A_i A_j$: In this case the number of A_j, which is definitely positive, will be decreased by one from $\vec{t_i}$.

(ii) $A_i \rightarrow A_i A_i$: In this case the number of A_i, which is definitely positive, will be decreased by one from $\vec{t_i}$.

From the above cases, we can conclude that $d(\vec{t_{i+1}}) = d(\vec{t_i}) - 1$ or $d(\vec{t_{i+1}}) < d(\vec{t_i})$.

(B) If a multiplicity vector $\vec{t_i}$ converts to a vector $\vec{t_{i+1}}$ under only the ascending rules of a given CHR(G), then, there are three possibilities for CFG rules (introduced in section 3.2):

(i) $A_i \rightarrow A_j A_k$: In this case the number of both A_j and A_k, which one of them is definitely positive, will be decreased by one, and A_i will be decreased by one, if both A_j and A_k be positive the $d(\vec{t_{i+1}}) = d(\vec{t_i}) - 1$, and if only one of them be positive, there will be no change, i.e. $d(\vec{t_{i+1}}) = d(\vec{t_i})$.

(ii) $A_i \rightarrow A_j A_j$: In this case the number of A_j, which is definitely positive, will be decreased by two, and A_i will be increased by one, then, if A_j be bigger than one, then, $d(\vec{t_{i+1}}) = d(\vec{t_i}) - 1$, and if the number of A_j be equal to one, then, $d(\vec{t_{i+1}}) = d(\vec{t_i})$.

(iii) $A_i \rightarrow A_iA_j$: In this case the number of A_j, which is definitely positive, will be decreased by one from $\vec{t_i}$, and the number of A_i will be increased by one, thus, $d(\vec{t_{i+1}}) = d(\vec{t_i})$.

From the above cases, we can conclude that $d(\vec{t_{i+1}}) \leq d(\vec{t_i})$.

A.5 Proof of the Property (E)

Property (E): A multiplicity vector $\vec{u}$ with zero degree has no applicable CHR rule.

Corollary: A reduction from a multiplicity vector $\vec{u}$ to a vector $\vec{v}$ with zero degree is terminated, since, $\vec{v}$ cannot be converted to any vector.

Proof:
Proof by contradiction. We assume there is a rule that can be applied. Thus, it shows that some nonterminals are positive, and this contradicts our assumption that $\vec{u}$ has no non-terminals with the positive occurrence.

A.6 Proof of the Property (F)

Property (F): For every $h \geq 1$, for every non-terminal x with positive number of occurrences in $\vec{u}$, and being reachable to a category y in h steps in a given proper CFG, G, there exists a sequence of reduction $\vec{u} = t_h, t_{h-1}, \ldots, t_0 = \vec{v}$ with length h, under a CHR(G), such that, the number of y in $\vec{v}$ is increased by one.

Proof:
We prove (F) by strong induction on h. We can assume (F) as P(h). Recall that the grammar is assumed to be in CNF.

I. Base Case: It should be shown that P(1) holds, i.e., for every non-terminal x with positive number of occurrences in $\vec{u}$, and being reachable to a category y in one step in a given proper CFG, G, there exists a sequence of reduction $\vec{u}(= t_1) \Rightarrow (t_0 =)\vec{v}$ with length one, under a CHR(G), such that, the number of y in $\vec{v}$ is increased by one.

Let X be a non-terminal and reachable to Y in one step, namely, $Y \Rightarrow \alpha X$ (or $Y \Rightarrow X\beta$) and the number of occurrence of X is more than one, thus, the corresponding CHR rule (for $Y \Rightarrow \alpha X$ or $Y \Rightarrow X\beta$) increases the number of Y by one, so P(1) holds. This shows that the base case holds.

II. Induction step: Assuming $P(i)$ is true for all $i \leq K$, it should be shown that $P(K+1)$ is true:

Let $\vec{u}(= t_{K+1})$ be a vector, and let X be a non-terminal with positive number of occurrences in $\vec{u}$, and being reachable to a category Y in $K+1$ steps in a given proper CFG, G. So, there is an X' such that X is reachable to X' in one step while X' is reachable to Y in K steps. So, there exists a CFG rule in the form of either $X' \Rightarrow \alpha X$ or $X' \Rightarrow X\beta$, so, let t_{K+1} be the result of corresponding CHR rule after applying to t_K, namely $t_{K+1} \rightarrow t_K$. (*) We can consider that the occurrence number of X' would be increased to one in t_K.

Based on inductive assumption, if X' be reachable to Y in K step with positive number of occurrence, then, there would exist a sequence of reduction $t_K, t_{K-1}, \ldots, t_0$ with length K, under a CHR(G), such that the number of Y in t_0 is increased by one.(**)

From (*) and (**) we can conclude:
There is a sequence $\vec{u}(= t_{K+1}), t_K, t_{K-1}, \ldots, t_0(= \vec{v})$ with length $K+1$, under a CHR(G), such that the number of Y in $\vec{v}$ is increased by one. This proves P(K+1), thus, induction step, i.e., $\forall i \leq K\ P(i) \Rightarrow P(K+1)$ holds.

III. Conclusion: Based on (I) and (II), and the principle of strong induction we can conclude that $\forall h \geq 1\ P(h)$ holds, i.e., property (F) holds.

A.7 Proof of the Property (G)

Property (G): There is a reduction sequence from any multiplicity vector $\vec{u}$ to a candidate vector $\vec{v}$ under a CHR(G) with G as proper CFG while $d(\vec{v}) < d(\vec{u})$.

Proof:
(G) is a corollary of (F):

Let X be a non-terminal with positive number of occurrences in $\vec{u}$, and being reachable to a category S in h steps in a given proper CFG, G, then, based on (F), there exists a sequence of reduction $\vec{u} = t_h, t_{h-1}, \ldots, t_0 = \vec{v}$ with length h, under a CHR(G), such that, the number of S in $\vec{v}$ is one. This proves that vector $\vec{v}$ is a candidate. Thus, there is a reduction sequence from any multiplicity vector $\vec{u}$ to a candidate vector $\vec{v}$. Based on (D), $d(\vec{t_{h-1}}) \leq d(\vec{h})$, so, $d(\vec{t_1}) \leq d(\vec{h})$ and $d(\vec{t_0}) < d(\vec{t_1})$ (because the number of S would not be counted in $d(\vec{t_0})$). We can conclude that $d(\vec{t_0}) < d(\vec{h})$ or $d(\vec{v}) < d(\vec{u})$.

A.8 Proof of the Property (H)

Property (H): For every h, under a proper CHR(G) and proper G, all possible sequence of reductions of $\vec{u}$ of degree $h = d(\vec{u})$ terminate.

We prove (H) by strong induction on h.

I. Base Case: It should be shown that H(0) holds, i.e., all possible sequence of reductions for $\vec{u}$ of degree zero, under a proper CHR(G) and proper G, terminate. This is obvious, since, based on (E), any vector with degree zero has no applicable rule, and this means termination. Thus, the base case holds.

II. Induction step: Assuming $H(i)$ is true for all $i \leq K$, it should be shown that $H(K+1)$ is true:

Let $\vec{u}$ be a vector with degree $k+1$, the are two possibilities:

(i) The occurrence number of S in $\vec{u}$ is zero: Since grammar G is proper, and based on (G), we have $\vec{u} \rightsquigarrow \vec{v}$ such that $\vec{v}$ is a candidate and $d(\vec{v}) < k+1$. Based on inductive assumption, $\vec{v}$ terminates, so, we can conclude that $\vec{u}$ terminates.

(ii) The occurrence number of S in $\vec{u}$ is positive:

Either a descending rule or an ascending rules can be applied to $\vec{u}$. If it is a descending rule, namely $\vec{u} \rightarrow \vec{u'}$, based on (D), $d(\vec{u'}) < k+1$. Based on inductive assumption, $\vec{u'}$ terminates, so, we can conclude that $\vec{u}$ terminates. If it is an ascending rule, the case is exactly as (i) above, and it terminates.

This proves H(k+1), thus, induction step, i.e., $\forall i \leq K\ H(i) \Rightarrow H(k+1)$ holds.

III. Conclusion: Based on (I) and (II), and the principle of strong induction we can conclude that $\forall h \geq 0\, H(h)$ holds, i.e., property (H) holds.

A.9 Proof of the Property (I)

Property (I): If $\vec{u} \rightsquigarrow \vec{t_h}$ ($\vec{u} = t_0, t_1, \ldots, t_h$), and $c_1, \ldots, c_h$ be the sequence of CHR(G) rules such that (for $i = 0, 1, \ldots, h-1$) t_{i+1} be conversion of t_i under c_{i+1} CHR(G) rule, then:

(A) There exists a procedure on the basis of the sequence $c_1, \ldots, c_h$ for building corrected parsed tree(s) of u.

(B) The elements in multiplicity vector t_h suggests the number of missing categories if it be negative; not-yet-used categories if be positive; and finally, zero indicates no-suggestion-so-far.

Proof:
We prove (I) by strong induction on h. Recall that the grammar is assumed to be in CNF.

I. Base Case: It should be shown that I(0) holds, since there is an empty sequence from $\vec{v}$, and there is nothing to do, thus, the base case holds.

II. Induction step: Assuming $I(i)$ is true for all $i \leq K$, it should be shown that $I(k+1)$ is true:

Let $\vec{u} \rightsquigarrow \vec{t_{k+1}}$ ($\vec{u} = t_0, t_1, \ldots, , t_k, t_{k+1}$) and $c_1, \ldots, c_k, c_{k+1}$ be the sequence of CHR(G) rules such that (for $i = 0, 1, \ldots, k$) t_{i+1} be conversion of t_i under c_{i+1} CHR(G) rule. CHR rule c_{k+1}, which corresponds to a CFG rule r_{k+1}, converts t_k to t_{k+1}. We can consider the following cases for r_{k+1} and in each case we show that (A) and (B) hold:

Ascending Rules of Type I:
(A) Build a new tree t by adjoining an existing tree t_1 with S as its root and A_i as its non-substitution node; and, t_2 as auxiliary tree, i.e. $A_i \rightarrow A_i A_j$ (or $A_i \rightarrow A_j A_i$), in which A_j in R.H.S is the foot node. t is obtained from adjoining t_2 to t_1 at non-substitution node A_i.

(B) The number of A_i is not changed since one A_i is used in L.H.S of r_{k+1} (indicates increasing by one) and one new rule is added in R.H.S (indicates decreasing by one). The number of A_j should be reduced by one, since A_j is used in R.H.S of r_{k+1}. We can see that the rule r_{k+1} truly keeps track of the number of the resources.

Note: More technical detail on tree adjoining techniques is available on TAG (=Tree Adjoining Grammar) literature. See [JS97; JS91, p.3].

Ascending Rules of Type II:
(A) Build a new tree t by adjoining an existing tree t_1 with S as its root and A_i as its non-substitution node; and t_2 as auxiliary tree, i.e. $A_i \rightarrow A_i A_i$ in which the first A_i in R.H.S of rule is the foot node. t is obtained from adjoining t_2 to t_1 at non-substitution node A_i.

(B) A_i is used in L.H.S of r_{k+1} (indicates increasing by one) and two new non-terminals is added in R.H.S (indicates decreasing by two). Thus, the number of A_i should be reduced by one. We can see that the rule r_{k+1} truly keeps track of the number of the resources.

Descending Rules of Type I:
(A) Build a new tree t by substitution of the existing trees (with roots of types A_j and A_k) with non-terminal nodes in R.H.S of rule $A_i \rightarrow A_i A_j$. In the end, t would have A_i as its root and substituted trees as its siblings.

(B) A_i is used in L.H.S of r_{k+1} (indicates increasing by one) and A_j and A_k is used in L.H.S of r_{k+1} (indicates increasing by one for each of them). We can see that the rule r_{k+1} truly keeps track of the number of the resources. If we use a non-terminal which does not have positive number then the number of occurrences might be negative which indicates missing non-terminals or something that is borrowed in order to complete the utterance.

Descending Rules of Type II:
(A) Build a new tree t by substitution of two existing trees (with roots of type A_j) with non-terminal nodes in R.H.S of rule $A_i \rightarrow A_j A_j$. In the end, t would have A_i as its root and substituted trees as its siblings.

(B) A_i is used in L.H.S of r_{k+1} (indicates increasing by one) and A_j is used two times in L.H.S of r_{k+1} (indicates increasing by two). We can see that the rule r_{k+1} truly keeps track of the number of the resources. If we use a non-terminal which does not have a positive number then the number of occurrences might be negative which indicates missing non-terminals or something that is borrowed in order to complete the utterance.

Descending Rules of Type III:
(A) Build a new tree t by substitution of existing tree (with root of type A_j) with non-terminal node A_j in R.H.S of rule $A_i \rightarrow A_i A_j$. In the end, t would have A_i as its root and A_j and substituted tree as its siblings.

(B) The number of A_i is not changed since one A_i is used in L.H.S of r_{k+1} (indicates increasing by one) and one new constraint is added with -1 for A_i (indicates decreasing by one). The number of A_j should be reduced by one since A_j is used in R.H.S of r_{k+1}. We can see that the rule r_{k+1} truly keeps track of the number of the resources.

By considering all the possible cases above, we can conclude that, induction step holds, i.e., $\forall i \leq k\ I(i) \Rightarrow I(k+1)$ holds.

III. Conclusion: Based on (I) and (II), and the principle of strong induction we can conclude that $\forall h \geq 0\, I(h)$ holds, i.e., property (I) holds.

A.10 Proof of the Property (J)

Property (J): If $\vec{u} \rightsquigarrow \vec{t_h}$ ($\vec{u} = t_0, t_1, \ldots, t_h = \vec{v}$), and $\vec{v}$ be a saturated candidate and $c_1, \ldots, c_h$ be the sequence of CHR(G) rules such that (for $i = 0, 1, \ldots, h-1$) t_{i+1} be conversion of t_i under c_{i+1} CHR(G) rule, then:

(A) There exists a procedure on the basis of the sequence $c_1, \ldots, c_h$ for building corrected parsed tree(s) of u.

(B) The elements in multiplicity vector t_h suggests the number of missing categories if it be negative;extra categories if it be positive; and finally, zero indicates no-suggestion.

Proof:
The proof can be straightforwardly gained by using the property (I) and an extra assumption that $\vec{v}$ is saturated candidate.

Example for the Property (J)

As mentioned, the procedure can be used as CHR rules for step-by-step generation of parsed trees. The yields of the parsed trees are the solution(s) as for incomplete sentences. We would work on the example 6.5 introduced in the chapter 6. For the convenience, it is repeated again.

Example A.1. *A girl helped the child admires the magician.

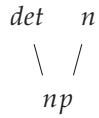
c[(cn,2), (det,2), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,1), (np′,0), (vp,0), (s,0),Ascending]

$$\begin{array}{c} det \quad n \\ \backslash \quad / \\ np \\ \\ t_1 \end{array}$$

c[(cn,1), (det,1), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,2), (np′,0), (vp,0), (s,0),Ascending]

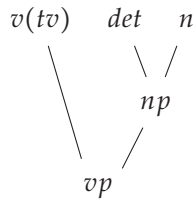
$$\begin{array}{c} det \quad n \\ \backslash \quad / \\ np \\ \\ t_2 \end{array}$$

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),2), (np,3), (np′,0), (vp,0), (s,0),Ascending]



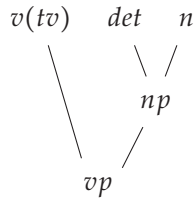
t_3

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),1), (np,2), (np′,0), (vp,1), (s,0),Ascending]



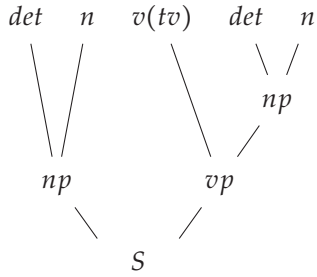
t_1

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),0), (np,1), (np′,0), (vp,2), (s,0),Ascending]



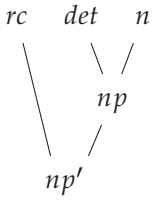
t_2

c[(cn,0), (det,0), (pn,0), (rc,0), (v(iv),0), (v(tv),0), (np,0), (np′,0), (vp,1), (s,1),Ascending]



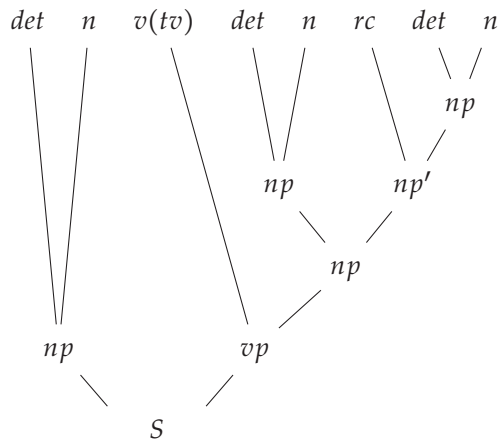
$t_2 + t_3$

c[(cn,0), (det,0), (pn,0), (rc,-1), (v(iv),0), (v(tv),0), (np,0), (np′,1), (vp,0), (s,1),Ascending]

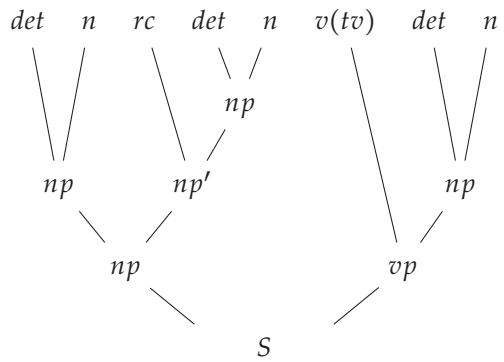


t_1

c[(cn,0), (det,0), (pn,0), (rc,-1), (v(iv),0), (v(tv),0), (np,0), (np′,0), (vp,0), (s,1),Descending]



$T_1 = t_1 + t_2 + t_3$



$T_2 = t_1 + t_2 + t_3$

We can observe that we have two trees, i.e. T_1 and T_2, in the end. This basically happened since we had two candidates *np* in the last constraint, thus, we have two possible updates. Now, let us write the yields of these trees once again here:

Solution (i): Yield of the tree T_1:
det,cn,v(tv),det,cn,rc,v(tv),det,cn

Solution (ii): Yield of the tree T_2:
det,cn,rc,v(tv),det,cn,v(tv),det,cn

We can see that our algorithm gives only two suggestions, namely, adding a relative clause to two different positions of the sentence. The solutions (i) and (ii) matches with our expectation:

Solution (i): A girl helped the child [who] admires the magician.
Solution (ii): A girl [who] helped the child admires the magician.

Our solution for fixation has its own drawbacks. This is discussed to some extent in the section 6.3.6.

Appendix B

Detailed Calculations of the Linguistic Complexity Metrics

This appendix provides the detailed calculations of different linguistic complexity metrics in different chapters of the dissertations. Recall that $ID_{\pi}(x)$ and $DL_{\pi}(x)$ are incomplete dependency number and dependency locality number, respectively. Also, $AccSum(x)$ indicates to the accumulative sum of the numbers.

B.1 Examples Related to the Chapter 3

3.3a	x	Everyone	repairs	something
	$ID(x)$	2	3	0
3.3b	y	Everyone	repairs	something
	$ID(y)$	4	3	0

TABLE B.1: Calculation of the incomplete dependency number for 3.3a and 3.3b.

3.6a	x	Everyone	repairs	something	expertly
	$ID(x)$	4	5	2	0
3.6b	y	Everyone	repairs	something	expertly
	$ID(y)$	4	3	2	0

TABLE B.2: Calculation of the incomplete dependency number for 3.6a and 3.6b.

B.2 Examples Related to the Chapter 5

5.3a	x	The	reporter	who	sent	the	photographer	to	the	editor	hoped	for	a	good	story
	$ID(x)$	3	4	4	4	4	3	3	3	2	1	1	1	1	0
5.3b	y	The	reporter	who	the	photographer	sent	to	the	editor	hoped	for	a	good	story
	$ID(y)$	3	4	4	6	5	3	3	3	2	1	1	1	1	0

TABLE B.3: Calculation of the incomplete dependency number for 5.3a and 5.3b.

5.3a	x	The	reporter	who	sent	the	photographer	to	the	editor	hoped	for	a	good	story
	$ID(x)$	0	0	1	1	0	0	0	0	0	5	0	0	0	0
	AccSum(x)	0	0	1	2	2	2	2	2	2	7	7	7	7	7
5.3b	y	The	reporter	who	the	photographer	sent	to	the	editor	hoped	for	a	good	story
	$ID(y)$	0	0	1	0	0	2	0	0	0	5	0	0	0	0
	AccSum(y)	0	0	1	1	1	3	3	3	3	8	8	8	8	8

TABLE B.4: Calculation of the dependency locality number for 5.3a and 5.3b.

5.4a	x	The	reporter	who	the	senator	who	John	met	attacked	disliked	the	editor
	$DL(x)$	0	0	1	0	0	1	0	2	4	6	0	0
	AccSum(x)	0	0	1	1	1	2	2	4	8	14	14	14
5.4b	y	The	reporter	who	the	senator	who	I	met	attacked	disliked	the	editor
	$DL(y)$	0	0	1	0	0	1	0	1	3	5	0	0
	AccSum(y)	0	0	1	1	1	2	2	3	6	11	11	11

TABLE B.5: Calculation of the dependency locality number for 5.4a and 5.4b.

5.5a	x	The	horse	raced	past	the	barn	
	$DL(x)$	0	0	0	2	0	0	0
	AccSum(x)	0	0	0	2	2	2	2
5.5b	y	The	horse	raced	past	the	barn	fell
	$DL(y)$	0	0	2	1	0	0	4
	AccSum(y)	0	0	2	3	3	3	7

TABLE B.6: Calculation of the dependency locality number for 5.5a and 5.5b.

5.6a	x	The	cheese	that	the	rat	that	the	cat	saw	ate	stank
	$DL(x)$	0	0	1	0	0	1	0	0	2	4	6
	AccSum(x)	0	0	1	1	1	2	2	2	4	8	14
5.6b	y	The	dog	that	chased	the	cat	that	saw	the	rat	barked
	$DL(y)$	0	0	1	1	0	0	1	1	0	0	6
	AccSum(y)	0	0	1	2	2	2	3	4	4	4	10

TABLE B.7: Calculation of the dependency locality number for 5.6a and 5.6b.

5.7a	x	Joe	said	that	Marthus	believes	that	Ingrid	fell	today
	$DL(x)$	0	2	0	0	2	0	0	0	2
	AccSum(x)	0	2	2	2	4	4	4	4	6
5.7b	y	Joe	said	that	Marthus	believes	that	Ingrid	fell	today
	$DL(y)$	0	2	0	0	0	0	0	2	4
	AccSum(y)	0	2	2	2	2	2	2	4	8
5.7c	z	Joe	said	that	Marthus	believes	that	Ingrid	fell	today
	$DL(z)$	0	0	0	0	2	0	0	2	6
	AccSum(z)	0	0	0	0	2	2	2	4	10

TABLE B.8: Calculation of the dependency locality number for the readings of 5.7.

5.8a	x	The	book	that	shocked	Mary	's	title
	$DL(x)$	0	0	1	1	0	3	0
	AccSum(x)	0	0	1	2	2	5	5
5.8b	y	The	book	that	shocked	Mary	's	title
	$DL(y)$	0	0	1	1	0	1	0
	AccSum(y)	0	0	1	2	2	3	3

TABLE B.9: Calculation of the dependency locality number for the readings of 5.8.

5.9a	x	Ingrid	was	astonished	that	Jack	was	surprised	that	two	plus	two	equals	four
	$DL(x)$	0	2	2	0	0	2	2	0	0	1	0	3	0
	AccSum(x)	0	2	4	4	4	6	8	8	8	9	9	12	12
5.9b	y	That	that	two	plus	two	equals	four	surprised	Jack	astonished	Ingrid		
	$DL(y)$	0	0	0	1	0	2	0	5	0	7	0		
	AccSum(y)	0	0	0	1	1	3	3	8	8	15	15		

TABLE B.10: Calculation of the dependency locality number for 5.9a and 5.9b.

B.3 Examples Related to the Chapter 6

6.6a	x	Every	child	fell	and	cried
	$ID(x)$	3	2	4	2	0
6.6b	y	Every	child	who	fell	cried
	$ID(y)$	3	4	4	2	0

TABLE B.11: Calculation of the incomplete dependency number for 6.6a and 6.6b.

6.6a	x	Every	child	fell	and	cried
	$DL(x)$	0	0	0	1	1
	AccSum(x)	0	0	0	1	2
6.6b	y	Every	child	who	fell	cried
	$DL(y)$	0	0	1	1	3
	AccSum(y)	0	0	1	2	5

TABLE B.12: Calculation of the dependency locality number for 6.6a and 6.6b.

6.6a	The	rat	that	the	cat	saw	died	
	3	4	4	6	5	2	0	
6.6b	The	rat	that	the	cat	briefly	saw	died
	3	4	4	6	5	5	2	0

TABLE B.13: Calculation of the incomplete dependency number for 6.7a and 6.7b.

6.6a	The	rat	that	the	cat	saw	died	
	0	0	1	0	0	2	4	
6.6b	The	rat	that	the	cat	briefly	saw	died
	0	0	1	0	0	1	2	4

TABLE B.14: Calculation of the dependency locality number for 6.7a and 6.7b.

6.6a	The	rat	that	the	cat	saw	died	
	3	4	4	6	5	2	0	
6.6b	The	rat	that	the	cat	briefly	saw	died
	3	4	2	3	2	2	2	0

TABLE B.15: Calculation of the activation-based complexity measurement for 6.7a and 6.7b.

B.4 Examples Related to the Chapter 7

7.1a	John	gave	Bill	the	painting	that	Mary	hated	
	2	2	1	1	2	2	3	0	
7.1b	John	gave	the	painting	that	Mary	hated	to	Bill
	2	2	2	3	3	4	1	1	0

TABLE B.16: Calculation of the incomplete dependency number for 7.1a and 7.1b.

7.1a	John 0	gave 2	Bill 0	the 0	painting 0	that 1	Mary 0	hated 2	
7.1b	John 0	gave 2	the 0	painting 0	that 1	Mary 0	hated 2	to 0	Bill 0

TABLE B.17: Calculation of the dependency locality number for 7.1a and 7.1b.

Appendix C

Published Work

This appendix provides the Ph.D. publications of the author and a quick review of their content. The core parts of the material describing preference modeling are the following references:

C.1 PhD Publications

The first published paper regarding the sentence completion (Algorithm A in chapter 6) was in TALN-Recital-2017:

- Mehdi Mirzapour, "Finding Missing Categories in Incomplete Utterances." In 24e Conférence sur le Traitement Automatique des Langues Naturelles (TAL), p. 149, 2017, Orléans, France.

The core parts of the material describing ranking quantifier scoping problem (chapter 3) was published in CONLI-2017 in this paper:

- Davide Catta, Mehdi Mirzapour, "Quantifier Scoping and Semantic Preferences." In 1st conference for Computing Natural Language Inference held by Centre for Linguistic Theory and Studies in Probability (joint with 12th International Conference on Computational Semantics), 2017, Montpellier, France.

The description of the collecting crowd-Sourced lexical coercions for compositional semantic (chapter 4) was first published in the LENLS-2017 and it was selected to be published in the Springer LNCS series in the post-proceeding volume:

- Mathieu Lafourcade, Bruno Mery, Mehdi Mirzapour, Richard Moot and Christian Retoré "Collecting Crowd-Sourced Lexical Coercions for Compositional Semantic Analysis" in Arai, S., Kojima, K., Mineshima, K., Bekki, D., Satoh, K., Ohta, Y. (Eds.) New Frontiers in Artificial Intelligence Springer LNCS 2018.

- Mathieu Lafourcade, Bruno Mery, Mehdi Mirzapour, Richard Moot and Christian Retoré, "Collecting Weighted Coercions from Crowd-Sourced Lexical Data for Compositional Semantic Analysis", Logic and Engineering of Natural Language Semantics 14 (LENLS 14), 2017, Tokyo, Japan.

The following paper published in LACompLing-2018 was the basis of the material in Chapter 5:

- Mehdi Mirzapour, Jean-Philippe Prost, Christian Retoré Categorial "Proof Nets and Dependency Locality: A New Metric for Linguistic Complexity" LACompLing 2018: Logic Algorithms in Computational Linguistics, Stockholm, August 28-31 2018.

C.2 Other Publications

A paper on diagrammatic logic published in Diagram-2018:

- Mehdi Mirzapour, Christian Retoré "Venn Diagram and Evaluation of Syllogisms with Negative Terms: A New Algorithm" in Chapman, P., Stapleton, G., Moktefi, A., Perez-Kriz, S., Bellucci, F. (Eds.) Diagrammatic Representation and Inference 10th International Conference, Diagrams 2018, Edinburgh, UK, June 18-22, 2018, LNCS / LNAI 10871, 2018.

- **Abstract:** This paper provides a quantitative computational account of why such a sentence has a harder parse than some other sentence, or that one analysis of a sentence is simpler than another one. We take for granted the Gibson's results on human processing complexity, and we provide a new metric which uses (Lambek) Categorial Proof Nets. In particular, we correctly model Gibson's account in his Dependency Locality Theory. The proposed metric correctly predicts some performance phenomena such as structures with embedded pronouns, garden pathing, unacceptability of center embedded, preference for lower attachment and passive paraphrases acceptability. Our proposal extends existing distance-based proposals on Categorial Proof Nets for complexity measurement while it opens the door to include semantic complexity, because of the syntax-semantics interface in categorial grammars.

Two papers on classical logic and sophistical refutation/semantics (written in Persian) are published in Logical Study Journal:

- Mahin Bagheri, Mehdi Mirzapour and Gholamreza Zakiani, "Medieval Supposition Theory: The Implicit Semantic of Aristotle's Sophistical Refutations", Logical Study, IHCS , Volume 9, Number 1, Spring 2018.

- **Abstract:** Supposition theory is one of the important logical semantic theories which is put forward by medieval logicians in their logical texts and commentaries under the discussion topic "Properties of Terms". Since this theory has important consequences in logic, philosophy and theology, we will investigate its conceptual and historical origin. We claim that there is a significant (historical and conceptual) bound between the medieval theory of supposition and Aristotle's theory of fallacies as he has stated in his treatise "sophistical refutations". The case-by-base study of Aristotle's fallacy in comparison to the semantical analysis of medieval logicians support the idea that supposition theory is the implicit semantic of Aristotle's "sophistical refutations" which has been reinterpreted as an explicit and dependent field of study by medieval logicians, and also it has been extended throughout the late medieval ages due to different semantical problems.

- Mohammad Hafi, Mahin Bagheri, Mehdi Mirzapour and Golamreza Zakiani, "The Groundability of Four Figures of Aristotelian's Syllogism" , Logical Study, IHCS , Volume 9, Number 2, Autumn 2018.

- **Abstract:** The purpose of this research is to provide a new concept in Aristotelian categorical syllogism which is called groundability. A valid mood is called groundable if one would derive all the 24 valid moods in the Aristotelian's syllogism by assuming only the valid mood and applying a chain of the following rules: simple conversion, reduction-ad-absurdum, subalternation, obversion and quantification negation. In this paper, we will prove that only the fifteen valid moods have the groundability property. Because

Aristotle proves all the valid moods of other figures based on the four moods in the first figure, he considers these moods of the first figure as moods having the groundability property. We show that the groundability is not restricted to the first valid moods of the first figure–they are fifteen moods as stated. Thus, it can be shown that Aristotle's purpose from the self-evidence of the first figure is not the groundability of the four moods in the first. This important logical result in Aristotle's system is gained through the introducing the concept of the groundability of the moods in syllogism. We show that unlike the common view in the Aristotelian tradition, it is not the case that the groundability of the first figure must be the basis for explaining of being self-evidence of the four moods of the first figure. Regardless of what lies behind the evidence of the first figure valid moods, this paper will eliminate one of the options which is somehow a common wrong interpretation for answering the problem.

Bibliography

[AD16] Ife Adebara and Veronica Dahl. "Grammar Induction as Automated Transformation between Constraint Solving Models of Language." In: *KnowProS@ IJCAI*. 2016.

[And+04] John R Anderson et al. "An integrated theory of the mind." In: *Psychological review* 111.4 (2004), p. 1036.

[Ash11] Nicholas Asher. *Lexical Meaning in Context: A Web of Words*. Cambridge University Press, Mar. 2011.

[Bek14] Daisuke Bekki. "Dependent Type Semantics: An Introduction". In: *Logic and Interactive RAtionality (LIRa) Yearbook 2012*. Ed. by Zoé Christoff et al. Vol. I. University of Amsterdam, 2014, pp. 277–300.

[Bev70] Thomas G Bever. "The cognitive basis for linguistic structures". In: *Cognition and the development of language* (1970).

[BJ11] Srinivas Bangalore and Aravind Joshi. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*. MIT Press, 2011.

[Bla00a] Philippe Blache. "Constraints, linguistic theories, and natural language processing". In: *International Conference on Natural Language Processing*. Springer. 2000, pp. 221–232.

[Bla00b] Philippe Blache. "Property grammars: A solution for parsing with constraints". In: *6th Int. Wks. on Parsing Technologies* (2000), pp. 295–296.

[Bla11a] Philippe Blache. "A computational model for linguistic complexity." In: *Proceedings of the first International Conference on Linguistics, Biology and Computer Science*. 2011.

[Bla11b] Philippe Blache. "Evaluating language complexity in context: New parameters for a constraint-based model". In: *CSLP-11, Workshop on Constraint Solving and Language Processing*. 2011.

[Bla16] Philippe Blache. "Representing syntax by means of properties: a formal framework for descriptive approaches". In: *Journal of Language Modelling* 4.2 (2016), pp. 183–224.

[Bon00] Roberto Bonato. "Uno studio sull'apprendibilità delle grammatiche di Lambek rigide—a study on learnability for rigid Lambek grammars". PhD thesis. Tesi di Laurea & Mémoire de DEA, Università di Verona & Université Rennes 1, 2000.

[BP08] Philippe Blache and Jean-Philippe Prost. "A Quantification Model of Grammaticality". In: *Proceedings of the Fifth International Workshop on Constraints and Language Processing (CSLP2008)*. to appear in Studies in Computational Intelligence, Springer. 2008. URL: `publis/blacheProst08-cslp.pdf`.

[BP14] Philippe Blache and Jean-Philippe Prost. "Constraints and Language". In: ed. by Philippe Blache et al. Cambridge Scholars Publishing, 2014. Chap. Model-Theoretic Syntax: Property Grammars, Status and Directions, pp. 37–59.

[BP90] Wojciech Buszkowski and Gerald Penn. "Categorial grammars determined from linguistic data by unification". In: *Studia Logica* 49.4 (1990), pp. 431–454.

[BR14] Roberto Bonato and Christian Retoré. "Learning Lambek grammars from proof frames". In: *Categories and Types in Logic, Language, and Physics*. Springer, 2014, pp. 108–135.

[Bus87] Wojciech Buszkowski. "Discovery procedures for categorial grammars". In: *Categories, Polymorphism and Unification* (1987), pp. 35–64.

[Cha+13] Jon Chamberlain et al. "Using Games to Create Language Resources: Successes and Limitations of the Approach". In: *The People's Web Meets NLP: Collaboratively Constructed Language Resources*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 3–44. ISBN: 978-3-642-35085-6. DOI: 10.1007/978-3-642-35085-6_1. URL: https://doi.org/10.1007/978-3-642-35085-6_1.

[Cha+15] Stergios Chatzikyriakidis et al. "Type Theories and Lexical Networks: Using Serious Games as the Basis for Multi-Sorted Typed Systems". In: *ESSLLI: European Summer School in Logic, Language and Information*. 2015.

[Cha+17a] S. Chatzikyriakidis et al. "Modern Type Theories and Lexical Networks: Using Serious Games as the Basis for Multi-Sorted Typed Systems". In: *Journal of Language Modelling* (2017).

[Cha+17b] Stergios Chatzikyriakidis et al. "An overview of Natural Language Inference Data Collection: The way forward?" In: *Proceedings of the Computing Natural Language Inference Workshop*. 2017.

[Cho14] Noam Chomsky. *Aspects of the Theory of Syntax*. Vol. 11. MIT press, 2014.

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax*. Cambridge: MIT Press, 1965.

[Cho82] Noam Chomsky. *Some concepts and consequences of the theory of government and binding*. Vol. 6. MIT press, 1982.

[CL14] Stergios Chatzikyriakidis and Zhaohui Luo. "Natural Language Inference in Coq". In: *J. of Logic, Lang. and Inf.* 23.4 (Dec. 2014), pp. 441–480. ISSN: 0925-8531. DOI: 10.1007/s10849-014-9208-x. URL: http://dx.doi.org/10.1007/s10849-014-9208-x.

[CM17] Davide Catta and Mehdi Mirzapour. "Quantifier Scoping and Semantic Preferences". In: *Proceedings of the Computing Natural Language Inference Workshop*. 2017.

[Coo07] Robin Cooper. "Copredication, dynamic generalized quantification and lexical innovation by coercion". In: *Fourth International Workshop on Generative Approaches to the Lexicon*. 2007.

[Coo83] Robin Cooper. "Quantification and Syntactic Theory, Reidel, Dordrecht". In: *CooperQuantification and Syntactic Theory1983* (1983).

[Cop+05] Ann Copestake et al. "Minimal recursion semantics: An introduction". In: *Research on Language and Computation* 3.2-3 (2005), pp. 281–332.

[CPR17] Stergios Chatzikyriakidis, Fabio Pasquali, and Christian Retoré. "From Logical and Linguistic Generics to Hilbert's tau and epsilon Quantifiers". In: *IfCoLog Journal of Logics and their Applications* 4.2 (2017), pp. 231–255.

[CQ69] Allan M Collins and M Ross Quillian. "Retrieval time from semantic memory". In: *Journal of verbal learning and verbal behavior* 8.2 (1969), pp. 240–247.

[Cru86] D. A. Cruse. *Lexical Semantics*. New York: Cambridge, 1986.

[CUM14] Philipp Cimiano, Christina Unger, and John McCrae. "Ontology-based interpretation of natural language". In: *Synthesis Lectures on Human Language Technologies* 7.2 (2014), pp. 1–178.

[Dav67] Donald Davidson. "The logical form of action sentences". In: (1967).

[DB04a] Verónica Dahl and Philippe Blache. "Directly executable constraint based grammars". In: *Proc. Journees Francophones de Programmation en Logique avec Contraintes, Angers, France*. 2004, pp. 149–166.

[DB04b] Verónica Dahl and Philippe Blache. *Implantation des grammaires de propriétés en CHR*. JFPLC-04, 2004.

[DB05] Veronica Dahl and Philippe Blache. "Extracting selected phrases through constraint satisfaction". In: *Proceedings of Constraint Solving and Language Processing (CSLP)*. Springer. 2005, pp. 3–17.

[DGR96] Philippe De Groote and Christian Retoré. "On the semantic readings of proof-nets". In: *Formal grammar 1996*. FoLLI. 1996, pp. 57–70.

[DGrt] Viviane Durand-Guerrier. *Négation et quantification dans la classe de mathématiques*. Forthcoming.

[Dik04] Alexander Dikovsky. "Dependencies as categories". In: *Proceedings of the Workshop on Recent Advances in Dependency Grammar*. Ed. by D. Duchier G-J. Kruiff. 2004.

[DM12] Veronica Dahl and J Emilio Miralles. "Womb grammars: Constraint solving for grammar induction". In: *Proceedings of the 9th Workshop on Constraint Handling Rules. vol. Technical report CW*. Vol. 624. 2012, pp. 32–40.

[DM93] Luca Dini and Giovanni Malnati. "Weak constraints and preference rules". In: *Studies in Machine Translation and Natural Language Processing* (1993), pp. 75–90.

[DPD09] Denys Duchier, Jean-Philippe Prost, and Thi-Bich-Hanh Dao. "A Model-Theoretic Framework for Grammaticality Judgements". In: *Proceedings of FG'09*. Vol. 5591. Springer, 2009. URL: `publis / DuchierProstDao09-FG09-MTSjudgements.pdf`.

[Frü95] Thom Frühwirth. "Constraint handling rules". In: *Constraint programming: Basics and trends*. Springer, 1995, pp. 90–107.

[Frü98] Thom Frühwirth. "Theory and practice of constraint handling rules". In: *The Journal of Logic Programming* 37.1-3 (1998), pp. 95–138.

[FW83] Dan Fass and Yorick Wilks. "Preference Semantics, Ill-formedness, and Metaphor". In: *Comput. Linguist.* 9.3-4 (July 1983), pp. 178–187. ISSN: 0891-2017. URL: `http://dl.acm.org/citation.cfm?id=1334.980082`.

[GDV08] Bruno Gaume, Karine Duvignau, and Martine Vanhove. "Semantic associations and confluences in paradigmatic networks". In: *From Polysemy to Semantic Change Towards a typology of lexical semantic associations, John Benjamins* (2008), pp. 233–264.

[Gib00] Edward Gibson. "The dependency locality theory: A distance-based theory of linguistic complexity". In: *Image, language, brain* (2000), pp. 95–126.

[Gib91] Edward Albert Fletcher Gibson. "A computational theory of human linguistic processing: Memory limitations and processing breakdown". PhD thesis. Carnegie Mellon University Pittsburgh, PA, 1991.

[Gib98] Edward Gibson. "Linguistic complexity: Locality of syntactic dependencies". In: *Cognition* 68.1 (1998), pp. 1–76.

[Gir11] Jean-Yves Girard. *The Blind Spot: lectures on logic*. European Mathematical Society, 2011.

[Gir87] Jean-Yves Girard. "Theoretical computer science". In: *Linear logic*. Vol. 50. 1987, pp. 1–102.

[GK98] E Gibson and K Ko. "An integration-based theory of computational resources in sentence comprehension". In: *Fourth Architectures and Mechanisms in Language Processing Conference, University of Freiburg, Germany*. 1998.

[GT96] Edwared Gibson and James Thomas. "The processing complexity of English center-embedded and self-embedded structures". In: *The proceedings of the North-Eastern Linguistic Society 1996*. Ed. by University of Massachusetts. 1996.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Vol. 7. Cambridge University Press Cambridge, 1989.

[Hal01] John Hale. "A probabilistic Earley parser as a psycholinguistic model". In: *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics. 2001, pp. 1–8.

[Hil22] David Hilbert. "Die logischen grundlagen der mathematik". In: *Mathematische Annalen* 88.1 (1922), pp. 151–165.

[Hop79] John E Hopcroft. *Introduction to Automata Theory, Languages and Computation: For VTU, 3/e*. Pearson Education India, 1979.

[HP+02] Rodney Huddleston, Geoffrey K Pullum, et al. "The cambridge grammar of english". In: *Language. Cambridge: Cambridge University Press* (2002), pp. 1–23.

[HS80] J Roger Hindley and Jonathan P Seldin. *To HB Curry: essays on combinatory logic, lambda calculus, and formalism*. Vol. 479490. Academic press London, 1980.

[HS87] Jerry R Hobbs and Stuart M Shieber. "An algorithm for generating quantifier scopings". In: *Computational Linguistics* 13.1-2 (1987), pp. 47–63.

[IL15] Seohyun Im and Chungmin Lee. "A Developed Analysis of Type Coercion Using Asher's TCL and Conventionality". In: *Extended abstracts of the ESSLLI 2015 workshop TYTLES: Types Theory and Lexical Semantics*. Ed. by Robin Cooper and Christian Retoré. Aug. 2015, pp. 91–99. URL: `https://hal.archives-ouvertes.fr/hal-01584832`.

[Jen+83] Karen Jensen et al. "Parse fitting and prose fixing: getting a hold on ill-formedness". In: *Computational Linguistics* 9.3-4 (1983), pp. 147–160.

[JK97] Aravind K Joshi and Seth Kulick. "Partial proof trees as building blocks for a categorial grammar". In: *Linguistics and Philosophy* 20.6 (1997), pp. 637–667.

[Joh98] Mark E. Johnson. "Proof Nets and the Complexity of Processing Center-Embedded Constructions". In: *Special Issue on Recent Advances in Logical and Algebraic Approaches to Grammar*. Ed. by C. Retoré. Vol. 7(4). Journal of Logic Language and Information. Kluwer, 1998, pp. 433–447.

[JS91] Aravind K Joshi and Yves Schabes. "Tree-adjoining grammars and lexicalized grammars". In: *Technical Reports (CIS)* (1991), p. 445.

[JS97] Aravind K Joshi and Yves Schabes. "Tree-adjoining grammars". In: *Handbook of formal languages*. Springer, 1997, pp. 69–123.

[Kan98] Makoto Kanazawa. *Learnable Classes of Categorial Grammars.* ERIC, 1998.

[Kel88] William R Keller. "Nested cooper storage: The proper treatment of quantification in ordinary noun phrases". In: *Natural language parsing and linguistic theories*. Springer, 1988, pp. 432–447.

[Kim73] John Kimball. "Seven principles of surface structure parsing in natural language". In: *Cognition* 2.1 (1973), pp. 15–47.

[Knu73] Donald E. Knuth. *The Art of Computer Programming, VoL 1: Fundamental Algorithms*. 1973.

[KPP14] Souhila Kaci, Namrata Patel, and Violaine Prince. "From NL preference expressions to comparative preference statements: A preliminary study in eliciting preferences for customised decision support". In: *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on*. IEEE. 2014, pp. 591–598.

[KR93] Hans Kamp and Uwe Reyle. "From Discourse to Logic; Introduction to the Modeltheoretic Semantics of natural language". In: (1993).

[Laf+p2] Mathieu Lafourcade et al. "Collecting Weighted Coercions from Crowd-Sourced Lexical Data for Compositional Semantic Analysis". In: *New Frontiers in Artificial Intelligence (JSAI-isAI 2017 Workshops, JURISIN, SKL, AI-Biz, LENLS, AAA, SCIDOCA, kNeXI, Revised Selected Papers)*. Ed. by Sachiyo Arai et al. Springer, TBP2018.

[Laf07] Mathieu Lafourcade. "Making people play for Lexical Acquisition with the JeuxDeMots prototype". In: *SNLP'07: 7th International Symposium on Natural Language Processing*. Pattaya, Chonburi, Thailand, Dec. 2007, p. 7. URL: `https://hal-lirmm.ccsd.cnrs.fr/lirmm-00200883`.

[Lam58] Joachim Lambek. "The mathematics of sentence structure". In: *The American Mathematical Monthly* 65.3 (1958), pp. 154–170.

[Lam97] Joachim Lambek. "Type grammar revisited". In: *International Conference on Logical Aspects of Computational Linguistics*. Springer. 1997, pp. 1–27.

[Las95] Alex Lascarides. "The Pragmatics of Word Meaning". In: *Proceedings of the AAAI Spring Symposium Series: Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity and Generativity*. 1995, pp. 75–80.

[LB17] Mathieu Lafourcade and Nathalie Le Brun. "Ambiguss, a game for building a Sense Annotated Corpus for French". In: *12th International Conference on Computational Semantics, Montpellier*. 2017.

[Lea+10] Claudia Leacock et al. "Automated grammatical error detection for language learners". In: *Synthesis lectures on human language technologies* 3.1 (2010), pp. 1–134.

[Lei17] Hans Leiß. "On Equality of Contexts and Completeness of the Indexed epsilon-Calculus". In: *IfCoLog Journal of Logics and their Applications* 4.2 (2017), pp. 347–366.

[LJ15] Mathieu Lafourcade and Alain Joubert. "TOTAKI: a help for lexical access on the TOT problem". In: *Language Production, Cognition, and the Lexicon*. Springer, 2015, pp. 95–112.

[Luo11] Zhaohui Luo. "Contextual analysis of word meanings in type-theoretical semantics". In: *LACL'11 Proceedings of the 6th international conference on Logical aspects of computational linguistics*. Springer-Verlag Berlin, 2011.

[Luo12] Zhaohui Luo. "Common Nouns as Types". English. In: *Logical Aspects of Computational Linguistics*. Ed. by Denis Béchet and Alexander Dikovsky. Vol. 7351. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 173–185. ISBN: 978-3-642-31261-8.

[LV05] Richard L Lewis and Shravan Vasishth. "An activation-based model of sentence processing as skilled memory retrieval". In: *Cognitive science* 29.3 (2005), pp. 375–419.

[MCP+95] Igor Mel, André Clas, Alain Polguère, et al. "Introduction à la lexicologie explicative et combinatoire". In: (1995).

[Mel89] Chris S Mellish. "Some chart-based techniques for parsing ill-formed input". In: *Proceedings of the 27th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1989, pp. 102–109.

[Mer17] Bruno Mery. "Challenges in the Computational Implementation of Montagovian Lexical Semantics". In: *New Frontiers in Artificial Intelligence: JSAI-isAI 2016 Workshops, LENLS, HAT-MASH, AI-Biz, JURISIN and SKL, Kanagawa, Japan, November 14-16, 2016, Revised Selected Papers*. Ed. by Setsuya Kurahashi et al. Cham: Springer International Publishing, 2017, pp. 90–107. ISBN: 978-3-319-61572-1.

[Mil95] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: `10.1145/219717.219748`. URL: `http://doi.acm.org/10.1145/219717.219748`.

[Mir] Mehdi Mirzapour. "Finding Missing Categories in Incomplete Utterances". In: *24e Conférence sur le Traitement Automatique des Langues Naturelles (TALN)*, p. 149.

[MMR15a] Bruno Mery, Richard Moot, and Christian Retoré. "Computing the Semantics of Plurals and Massive Entities using Many-Sorted Types". In: *New Frontiers in Artificial Intelligence JSAI-isAI 2014 Workshops, LENLS, JURISIN, and GABA, Kanagawa, Japan, October 27-28, 2014, Revised Selected Papers*. Ed. by Tsuyoshi Murata, Koji Mineshima, and Daisuke Bekki. Vol. 9067. Lecture Notes in Artificial Intelligence. Springer-Verlag Berlin Heidelberg, 2015, p. 357. DOI: `10.1007/978-3-662-48119-6`. URL: `https://hal.inria.fr/hal-01214435`.

[MMR15b] Bruno Mery, Richard Moot, and Christian Retoré. "Typed Hilbert Operators for the Lexical Semantics of Singular and Plural Determiner Phrases". In: *Epsilon 2015 – Hilbert's Epsilon and Tau in Logic, Informatics and Linguistics*. Montpellier, France, June 2015.

[Moo02] Richard Moot. "Proof nets for linguistic analysis". PhD thesis. UIL-OTS, Universiteit Utrecht, 2002.

[Moo10] Richard Moot. "Wide-Coverage French Syntax and Semantics using Grail". In: *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*. System Demo. Montreal, 2010.

[Moo12] Richard Moot. "Wide-Coverage Semantics for Spatio-Temporal Reasoning". In: *Traitement Automatique des Languages* 53.2 (2012), pp. 115–142.

[Moo15] Richard Moot. "A Type-logical Treebank for French". In: *Journal of Language Modelling* 3.1 (2015), pp. 229–264.

[Moo17] Richard Moot. "The Grail Theorem Prover: Type Theory for Syntax and Semantics". In: *Modern Perspectives in Type-Theoretical Semantics*. Ed. by Stergios Chatzikyriakidis and Zhaohui Luo. Cham: Springer International Publishing, 2017, pp. 247–277. ISBN: 978-3-319-50422-3. DOI: 10.1007/978-3-319-50422-3_10. URL: https://doi.org/10.1007/978-3-319-50422-3_10.

[Mor00] Glyn Morrill. "Incremental processing and acceptability". In: *Computational linguistics* 26.3 (2000), pp. 319–338.

[MPR18] Mehdi Mirzapour, Jean-Philippe Prost, and Christian Retoré. "Categorial Proof Nets and Dependency Locality: A New Metric for Linguistic Complexity". In: *Logic and Algorithms in Computational Linguistics*. Stockholm University. 2018.

[MR12a] Richard Moot and Christian Retoré. *The logic of categorial grammars: a deductive account of natural language syntax and semantics*. Anglais. Vol. 6850. LNCS. Springer, 2012. URL: http://www.springer.com/computer/theoretical+computer+science/book/978-3-642-31554-1.

[MR12b] Richard Moot and Christian Retoré. *The logic of categorial grammars: a deductive account of natural language syntax and semantics*. Vol. 6850. Springer, 2012.

[MR13] Bruno Mery and Christian Retoré. "Semantic types, lexical sorts and classifiers". In: *arXiv preprint arXiv:1312.3168* (2013).

[MR15] Bruno Mery and Christian Retoré. "Are books events ? Ontological Inclusions as Coercive Sub-Typing, Lexical Transfers as Entailment". In: *LENLS '12, in jSAI 2015*. Kanagawa, Japan, Nov. 2015.

[Mus90] Reinhard Muskens. "Anaphora and the logic of change". In: *European Workshop on Logics in Artificial Intelligence*. Springer. 1990, pp. 412–427.

[Nag94] Katashi Nagao. "A preferential constraint satisfaction technique for natural language analysis". In: *IEICE TRANSACTIONS on Information and Systems* 77.2 (1994), pp. 161–170.

[Nij80] Anton Nijholt. *Context-free grammars: covers, normal forms, and parsing*. 93. Springer Science & Business Media, 1980.

[Par90] Terence Parsons. *Events in the Semantics of English*. Vol. 5. Cambridge, Ma: MIT Press, 1990.

[Pat16] Namrata Patel. "Preference Handling in Decision-Making Problems". PhD thesis. Université de Montpellier, 2016.

[PL11] Jean-Philippe Prost and Mathieu Lafourcade. "Pairing Model-Theoretic Syntax and Semantic Network for Writing Assistance". In: *Proceedings of the 6th International Workshop on Constraints and Language Processing (CSLP@Context'11)*. 2011. URL: `publis/prostLafourcade2011-cslp.pdf`.

[Pol06] Alain Polguère. "Structural properties of lexical systems: Monolingual and multilingual perspectives". In: *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*. Association for Computational Linguistics. 2006, pp. 50–59.

[Pro08] Jean-Philippe Prost. "Modelling Syntactic Gradience with Loose Constraint-based Parsing". alternative url: http://hdl.handle.net/1959.14/28841. PhD thesis. Macquarie University, Sydney, Australia, and Universitï¿½ de Provence, Aix-en-Provence, France (cotutelle), 2008. URL: `http://tel.archives-ouvertes.fr/tel-00352828/fr/`.

[Pro10] Jean-Philippe Prost. *Graded Grammaticality: a Computational Framework*. in press. LAP Lambert Academic Publishing, 2010.

[PS01] Geoffrey K Pullum and Barbara C Scholz. "On the distinction between model-theoretic and generative-enumerative syntactic frameworks". In: *International Conference on Logical Aspects of Computational Linguistics*. Springer. 2001, pp. 17–43.

[Pus91] James Pustejovsky. "The generative lexicon". In: *Computational linguistics* 17.4 (1991), pp. 409–441.

[Pus95] James Pustejovsky. *The Generative Lexicon*. MIT Press, 1995.

[Res67] Nicholas Rescher. *The logic of decision and action*. University of Pittsburgh Pre, 1967.

[Ret14] Christian Retoré. "The Montagovian Generative Lexicon Lambda Ty_n: a Type Theoretical Framework for Natural Language Semantics". In: *19th International Conference on Types for Proofs and Programs (TYPES 2013)*. Vol. 26. Leibniz International Proceedings in Informatics (LIPIcs). Germany: Schloss Dagstuhl, 2014, pp. 202–229. ISBN: 978-3-939897-72-9.

[Ret96] Christian Retoré. "Calcul de Lambek et logique linéaire". In: *Traitement Automatique des Langues* 37.2 (1996), pp. 39–70.

[Roo91] Dirk Roorda. "Resource logic: proof theoretical investigations". PhD thesis. FWI, Universiteit van Amsterdam, 1991.

[Roo92] Dirk Roorda. "Proof nets for Lambek calculus". In: *Logic and Computation* 2.2 (1992), pp. 211–233.

[Sad08] Mehrnoosh Sadrzadeh. "Pregroup Analysis of Persian Sentences". In: *Recent Computational Algebraic Approaches to Morphology and Syntax, Polimetrica, Milan, 2008.* Ed. by C Casadio and J Lambek. 2008.

[SM98] David Schneider and Kathleen F McCoy. "Recognizing syntactic errors in the writing of second language learners". In: *Proceedings of the 17th international conference on Computational linguistics-Volume 2*. Association for Computational Linguistics. 1998, pp. 1198–1204.

[Ste12] Mark Steedman. *Taking scope: The natural semantics of quantifiers*. MIT Press, 2012.

[Ste99] Mark Steedman. "Alternating quantifier scope in CCG". In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics. 1999, pp. 301–308.

[Szy16] Jakub Szymanik. *Quantifiers and cognition: Logical and computational perspectives*. Vol. 96. Springer, 2016.

[Van78] Kurt VanLehn. "Determining the Scope of English Quantifiers". AI-TR-483, Artificial Intelligence Laboratory. MA thesis. Cambridge, MA: MIT, 1978.

[Vas+05] Shravan Vasishth et al. "Quantifying Processing Difficulty in Human Language Processing". In: *In Rama Kant Agnihotri and Tista Bagchi* (2005).

[VC95] Carl Vogel and Robin Cooper. "Robust chart parsing with mildly inconsistent feature structures". In: *Nonclassical feature systems* 10 (1995), pp. 197–216.

[VEU10] Jan Van Eijck and Christina Unger. *Computational semantics with functional programming*. Cambridge University Press, 2010.

[WG99] Tessa Warren and Edward Gibson. "The effects of discourse status on intuitive complexity: Implications for quantifying distance in a locality-based theory of linguistic complexity". In: *Poster presented at the Twelfth CUNY Sentence Processing Conference, New York*. 1999.