AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Institut National de Recherche en Informatique et Automatique — Université de Lorraine
École doctorale IAEM Informatique, Automatique, Électronique et Mathématiques

Expression de la dynamique du discours à l'aide de continuations

THÈSE

présentée et soutenue publiquement le 6 Avril 2012

pour l'obtention du

Doctorat de l'Université de Lorraine

(spécialité informatique linguistique)

par

Ekaterina LEBEDEVA

Composition du jury

Rapporteurs :	Nicholas ASHER	Directeur de recherche, CNRS, Toulouse, France
	Reinhard MUSKENS	Professeur, Tilburg University, Pays-Bas
Examinateurs :	Paul EGRÉ	Chargé de recherche, CNRS, Paris, France
	Jean-Marie PIERREL	Professeur, Université de Lorraine, Nancy, France
Directeur de thèse :	Philippe DE GROOTE	Directeur de recherche, INRIA, Nancy, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Abstract

This thesis develops a theoretical formalism of formal semantics of natural language in the spirit of Montague semantics. The developed framework satisfies the principle of compositionality in a simple and elegant way, by being as parsimonious as possible: completely new formalisms or extensions of existing formalisms with even more complex constructions to fit particular linguistic phenomena have been avoided; instead, the framework handles these linguistic phenomena using only basic and well-established formalisms, such as simply-typed lambda calculus and classical logic. Dynamics is achieved by employing a continuation-passing technique and an exception raising and handling mechanism. The context is explicitly represented by a term, and, therefore, can be easily accessed and manipulated. The framework successfully handles cross-sentential anaphora and presuppositions triggered by referring expressions and has potential to be extended for dealing with more complex dynamic phenomena, such as presuppositions triggered by factive verbs and conversational implicatures.

Keywords: computational linguistics, natural language semantics, Montague semantics, dynamic semantics, continuations, exceptions, presuppositions, conversational implicatures.

Acknowledgments

Two courses that I followed during my MSc in Computational Logic influenced me to do a PhD in the area of computational linguistics. The first course was an introduction to computational linguistics taught by Dr. Rafaella Bernardi at Free University of Bozen-Bolzano, Italy, where I learned how logical methods can be used to solve many linguistic problems elegantly. Dr. Bernardi's enthusiasm about the topic and her expertise inspired me and raised my interest in the area. The other course was an introduction to lambda calculus given by Prof. Alexander Leitsch at Vienna University of Technology, Austria. The course was very challenging, demanding and taught me to be mathematically precise, concise, creative and productive. On a seminar related to the lambda calculus course, I gave a talk presenting the basics of Montague's semantics. I was so fascinated with what I learned for the talk that I decided that if I ever did a PhD, this should be the area of my work.

I was very lucky to reach this decision exactly when Prof. Philippe de Groote announced an open PhD position at Loria (Laboratoire Lorrain de Recherche en Informatique et ses Applications), INRIA, Nancy, France, on extending Montague's semantics with continuation techniques. Prof. de Groote gave me an opportunity to work in his challenging project in the intersection of exciting areas such as logic, lambda calculus, natural language semantics and pragmatics, and established the theoretical and methodological foundations for my research. He has a fantastic ability of supervising the work concisely and precisely, by giving just enough advice based on the student's background and simultaneously intriguing and challenging the student to go ahead and investigate the problem more deeply. This kind of supervision provides an excellent scientific environment for the development of a proper research spirit. Prof. de Groote systematically guided my research and was demanding to my primary work, but at the same time he gave me enough freedom to explore. His openness to ideas and affirmative critics helped me stay on track and to form my own perception of the subject.

Prof. Nicholas Asher and Prof. Reinhard Muskens accepted to be the reviewers of my thesis. Dr. Paul Egré and Prof. Jean-Marie Pierrel accepted to be the members of the thesis committee. I appreciate their time and effort for reading the thesis. Their reviews and questions gave me a significant constructive feedback.

The CAuLD (Construction Automatique de représentations Logiques du Discours) workshops, organized by Dr. Sylvain Pogodalla, were essential at the beginning of my PhD. They gave me a community to interact with and from whom to obtain feedback. They also gave me the opportunity to practice important skills such as presenting my work.

I had chances to meet inspiring researchers during the years of my PhD. Lectures in ESSLLIs (European Summer School of Logic, Language and Information), talks in conferences or workshops and informal conversations with Prof. Nicholas Asher, Prof. Chris Barker, Prof. Makoto Kanazawa, Prof. Reinhard Muskens and Dr. Sylvain Salvati substantially influenced my work.

Prof. Patrick Blackburn was a source of positive thinking for me with respect to not only research difficulties, but also life troubles. He was magically appearing around when I needed support, a wise advice or just cheering up.

Dr. Luciana Benotti's enthusiasm, persistence and affection to research were an example for me to follow. She motivated me in numerous ways by being not only an inspiring colleague but also a reliable friend.

The harmonious and friendly environment of the Calligramme/Semagramme team (Dr. Maxime Amblard, Dr. Daniel de Carvalho, Prof. Philippe de Groote, Dr. Bruno Guillaume, Mag. Robert Hein, Prof. François Lamarche, Mag. Sarah Maarek, Mag. Jonathan Marchand, Dr. Mathieu Morey, Dr. Novak Novaković, Prof. Guy Perrier, Dr. Sylvain Pogodalla, Mag. Florent Pompigne, Mag. Shohreh Tabatabayi, Mag. Sai Qian) allowed me to stay tranquil and focused at work. I also met wonderful, kind and supportive people working in the Talaris team (Mag. Corinna Anderson, Dr. Carlos Areces, Dr. Luciana Benotti, Prof. Patrick Blackburn, Dr. Alexandre Denis, Dr. Ingrid Falk, Prof. Claire Gardent, Dr. Sébastien Hinderer, Dr. Guillaume Hoffmann, Prof. Jean-Charles Lamirel, Mag. Alejandra Lorenzo, Dr. Yannick Parmentier, Mag. Laura Pérez, Dr. Lina Maria Rojas, Dr. Dmitry Sustretov). I loved to spend lunch breaks with the great company of the members of the Mosel/VeriDis team (Dr. Nazim Benaissa, Dr. Diego Caminha, Mag. Henri Debrat, Dr. Pascal Fontaine, Dr. Neeraj Kumar, Mag. Tianxiang Lu, Prof. Dominique Méry, Prof. Stephan Merz, Dr. Denis Roegel, Mag. Hernán Vanzetto, Dr. Bruno Woltzenlogel Paleo) having pleasant conversations about research, education, travelling, sports, politics and many other interesting topics.

Staying late for work in the laboratory was encouraging, because Denis Roegel was always there and open for having a short break and a nice conversation. He offered me so much help when I needed, from providing valuable advice related to the French language and LaTeX to giving a hand to solve housing problems.

There was a place in Nancy where I felt as being at home. The family of Claire Gardent (Jennifer, Gabrielle and Caroline) was always welcoming me and was treating me as a family member. Before leaving Nancy, I spent so many evenings in their warm family atmosphere; after having left Nancy, I was welcomed to stay in their home numerous times when I came back for work. Jennifer supported me very much in learning French. Claire, moreover, introduced the world of yoga to me. Her Iyengar yoga classes at Loria helped me to stay in the right physical and mental shape during the years of PhD. Now yoga has a special place in my life.

I had great leisure time learning French, doing sports, dancing, cooking, having dinners, parties and generally enjoying life in Nancy thankful to the friendship of Carlos Areces, Luciana Benotti, Jennifer Blackburn, Patrick Blackburn, Guillermo Caminer, Diego Caminha, Pascal Fontaine, Claire Gardent, Alejandra Lorenzo, Atif Mashkoor, Stephan Merz, Novak Novaković, Denis Roegel, Sai Qian and Nadiya Yampolska.

This thesis is a culmination of all the efforts my family, parents and grandparents, spent for my education. They devoted all their resources and did their best to create good conditions for my studies. They taught me to be persistent and diligent when working towards goals, and this quality was essential for writing this thesis. It is dedicated to them.

From the first days of my PhD, Bruno Woltzenlogel Paleo had no choice but to hear me out talking about my research and listen to all my scientific thoughts regardless if it was during a vacation walk along a beach or a romantic dinner. Astonishingly, he seemed to like this so much that he sometimes initiated the scientific discussion himself. Eventually, he decided to marry me when I was halfway through my PhD, and his family warmly accepted me. Bruno's confidence in me, encouragement, patience and love kept me going forward in research and life.

I am thankful to everyone mentioned above. Without their support and advice this thesis would not have existed.

Contents

1 Formal Semantics of Natural Language **5**
- 1.1 Truth-Conditionality . . . 5
- 1.2 Compositionality . . . 8
- 1.3 Model-Theory . . . 11
- 1.4 Montague Semantics . . . 22
- 1.5 Abstract Categorial Grammars . . . 29

2 Natural Language Dynamics **41**
- 2.1 Motivation for a Dynamic Theory . . . 41
  - 2.1.1 Context . . . 43
  - 2.1.2 Cross-Sentential Pronominal Anaphora . . . 46
  - 2.1.3 Donkey Sentences . . . 49
  - 2.1.4 Presuppositions . . . 53
  - 2.1.5 Conversational Implicatures . . . 61
- 2.2 Dynamic Approaches . . . 62
  - 2.2.1 Discourse Representation Theory . . . 62
  - 2.2.2 Dynamic Predicate Logic . . . 70
  - 2.2.3 Dynamic Montague Grammar . . . 72

3 Continuation **81**
- 3.1 CPS Transformation . . . 82
- 3.2 CPS Control . . . 84
- 3.3 Continuation Technique in Natural Language Semantics . . . 90

4 Continuation-based Dynamic Logic **93**
- 4.1 Framework G_0: Initial Approach . . . 94
- 4.2 Framework G . . . 106
  - 4.2.1 Formal Details . . . 106

4.2.2 Donkey Sentences . . . 112
4.3 Framework GL . . . 114
4.3.1 Formal Details . . . 114
4.3.2 Linguistic Motivations . . . 119
4.3.3 Conservation . . . 123

5 From Static to Dynamic Meaning 143
5.1 Dynamic Interpretations Analogous to Static Interpretations . . . 144
5.1.1 Nouns . . . 144
5.1.2 Conventional Verbs . . . 145
5.1.3 Indefinite Article . . . 146
5.1.4 Determiner "every" . . . 147
5.1.5 Relative Pronouns . . . 148
5.2 Referring Expressions as Exception Triggers . . . 148
5.2.1 Selection Function and Exceptions . . . 149
5.2.2 Proper Names . . . 150
5.2.3 From Proper Names to Pronouns . . . 152
5.2.4 Definite Article . . . 154
5.2.5 Genitives . . . 156
5.3 Comparison of Dynamic and Static Lexical Interpretations . . . 158
5.4 Sentence Meaning . . . 160
5.5 Donkey Sentences . . . 166

6 Continuation-based Dynamic Logic with Exceptions 179
6.1 Discourse Update: Accommodation as Exception Handling . . . 179
6.2 Framework GLχ . . . 186
6.3 Cross-Sentential Anaphora . . . 203
6.4 Presupposition Projection in Conditionals . . . 206
6.5 Binding Problem . . . 224

7 Directions for Further Development of the Framework 231
7.1 Presuppositions . . . 231
7.1.1 Factive Verbs . . . 231
7.1.2 Additional exception handler in discourse update . . . 237
7.2 Conversational Implicatures . . . 240
7.2.1 Conversational Implicatures by Proof-Theoretical Abduction 240
7.2.2 Conversational Implicatures by Correcting the Context . . . 243
7.2.3 Additional exception handlers in discourse update . . . 246
7.2.4 Simplifying Dynamization Function . . . 247
7.3 Presuppositions vs. Conversational Implicatures . . . 247

8 Conclusions **259**
8.1 Summary of Results 259
8.2 Comparison with Related Work 260
8.2.1 Discourse Representation Theory 260
8.2.2 Dynamic Predicate Logic 261
8.2.3 Dynamic Montague Grammar 265
8.3 Advantages of the Developed Framework 268
8.4 Future Work 269

A Lambda Calculus **271**
A.1 Type-free Lambda Calculus 271
A.2 Simply-typed Lambda Calculus 276

Bibliography **281**

Introduction

Semantics is the study of meaning. Natural language semantics is the study of language meaning. This dissertation focuses on formal semantics of natural language, which seeks to express the meaning of natural language expressions with mathematically precise methods.

The development of formal natural language semantics is important in, at least, two aspects. From a scientific point of view, it deepens our knowledge of our own rich and complex languages and provides answers to interesting questions such as how human beings are capable of understanding potentially infinite number of complex expressions including expressions that they have not heard before. From a technological point of view, natural language semantics is in the core of natural language processing techniques and applications such as machine translation, dialogue systems, textual entailment, reasoning with unstructured knowledge expressed in natural language, question answering and information extraction. Expressing meanings of natural language expressions more precisely and with simpler formalisms can significantly improve these techniques and applications.

However, due to the complexity of natural language, obtaining the meaning of natural language expressions by formal means is a very difficult task. The goal of this dissertation is to present a framework that aims at facilitating this task and being as expressive and yet simpler than previous approaches.

One of the major advances in interpreting natural language formally was the work of Richard Montague [1970a; 1970b; 1973], who argued that natural language meaning can be compositionally computed with precise mathematical methods. He presented a formal framework capable of computing the meanings of independent sentences of a fragment of English. The fundamentals of Montague semantics are discussed in Section 1.4.

However, Montague's framework, as presented in [1970a; 1970b; 1973], was limited to single and relatively simple sentences, and could not handle complex phenomena such as, for example, cross-sentential and donkey anaphora (Sec-

tions 2.1.2 and 2.1.3 respectively).

In order to cope with the limitations of Montague's framework, various theories, aimed specifically on **discourse dynamics** were developed, including discourse representation theory (DRT) [Kamp, 1981; Kamp and Reyle, 1993], file change semantics (FCS) [Heim, 1982; Heim, 1983], dynamic predicate logic (DPL) [Groenendijk and Stokhof, 1991], dynamic montague grammar (DMG) [Groenendijk and Stokhof, 1990] and their extensions [Chierchia, 1992; Asher, 1993; Muskens, 1996; Dekker, 1999].

Although these frameworks definitely helped in understanding many problems related to discourse dynamics, they are not completely satisfactory (as discussed in Chapter 2). Some of them deviate from Montague's requirement of compositionality (Section 1.2), while others attempt to stay within Montague's theory but then either use non-standard logics or introduce additional syntactic structures into their logical languages, and thus compromise their simplicity and elegance.

In contrast, the framework presented in this dissertation satisfies the principle of compositionality in a simple and elegant way, by being as parsimonious as possible: completely new formalisms or extensions of existing formalisms with complex constructions to fit particular linguistic phenomena have been avoided; instead, the framework handles these linguistic phenomena using only basic and well-established formalisms, such as simply-typed lambda calculus and classical logic. The goal has always been to do more with less. This required a careful investigation of how discourse dynamics could be expressed with standard mathematical tools [de Groote, 2006], which later culminated in the development of a dynamic logic based on the notion of continuation. The framework is defined in Chapter 4 and is further extended in Chapters 5, 6 and 7.

This work is organized according to the following chapters:

Chapter 1, *Formal Semantics of Natural Language*, briefly discusses fundamentals necessary for understanding formal semantics of natural language in general, including Truth-conditionality, compositionality, model-theory, Montague semantics and abstract categorial grammars.

Chapter 2, *Natural Language Dynamics*, talks about phenomena of discourse dynamics, such as cross-sentential pronominal and donkey anaphora, presuppositions and conversational implicatures, and overviews the currently most used theories, namely Discourse Representation Theory, Dynamic Predicate Logic and Dynamic Montague Grammar, that are designed to cope with discourse dynamics.

Chapter 3, *Continuation*, presents basic principles and examples of the continuation passing technique developed to formalize dynamics in mathematical semantics of programming languages in a compositional way.

Chapter 4, *Continuation-based Dynamic Logic*, applies the continuation passing technique in the formalization of discourse dynamics. It defines a systematic translation of familiar static terms of Montague's semantics into their dynamic equivalents. Importantly, a systematic translation of non-logical constants is defined. A conservation result saying that a proposition is true in a model if and only if its dynamic variant is true in the same model is proved.

Chapter 5, *From Static to Dynamic Meaning: Interpretation of Lexical Items and Sentences*, presents how the usual static lexical interpretations for words of various syntactic categories are translated into dynamic interpretations. The special treatment of context allows the framework to interpret lexical items with potential side effects by using an exception raising technique, as demonstrated for referring expressions. Moreover, it is shown that dynamic lexical interpretations can be represented compactly and in a way that is structurally analogous to their static counterparts.

Chapter 6, *Continuation-based Dynamic Logic with Exceptions: Interpretation of Discourse*, presents how meanings of discourses can be compositionally computed. It defines a discourse update function that, when given interpretations of a discourse and a sentence, returns an interpretation of a new discourse. An exception handling mechanism is used for accommodating familiarity presuppositions triggered by referring expressions if the context of the initial discourse does not contain appropriate referents. The use of exceptions required an extension of the framework's calculus.

Chapter 7, *Directions for Further Development of the Framework*, discusses potential extensions of the framework, namely some phenomena related to presuppositions and conversational implicatures and, thereby, illustrates the possibility to handle semantic and (some) pragmatic phenomena in a unified framework.

Chapter 8, *Conclusions*, briefly compares the developed framework with related work and summarizes the results.

Chapter 1
Formal Semantics of Natural Language

This chapter sets the foundations for semantics of natural language. First, the basics of model-theoretical interpretation of a language are described, starting from Tarski's truth-conditional semantics and an informal explanation what an interpretation from an object language to a meta-language (in which the meaning is represented) is; proceeding with a notion of compositionality, which guarantees that the interpretation from the object language into the meta-language is adequate; and, finally, providing the fundamentals of model-theory to which the truth-conditional semantics evolved over time. Model-theoretical interpretations of interesting sentences like *The Morning star is the Evening star* and *The king of France is bald* in different models illustrate that the truth value of a sentence is relative to a model in which it is interpreted. Afterwards, the basic ideas of Montague semantics, which is a theory of compositional interpretation of natural language, are presented. The final section talks about the relation between syntax and semantics of a language and presents it using the formalism called abstract categorial grammars.

As notions of language and grammar are important throughout this and the following chapters, they are defined below:

Definition 1.1. [Language] A **language** is a set of finite strings (sentences) of symbols from an alphabet.

Definition 1.2. [Grammar] A **grammar** is a set of rules indicating how symbols of an alphabet can be combined to form expressions. A grammar specifies the structure of expressions.

1.1 Truth-Conditionality

Truth-conditional semantics has its roots in the work of Alfred Tarski, who formulated his classic theory of truth in 1933 in "The Concept of Truth in Formalized

Languages" [Tarski, 1956]. Even though the article contains a section named "The concept of true sentence in everyday or colloquial language" and devoted to defining truth of natural language sentences, it is merely used "for the purpose of introducing the reader to our subject". The theory itself was originally formulated for artificial languages of mathematical logic only. Tarski was very sceptical to the possibility of defining such a theory for natural languages. His concerns were related to universality and ambiguity of natural language:

> [...] the concept of truth (as well as other semantical concepts) when applied to colloquial language in conjunction with the normal laws of logic leads inevitably to confusions and contradictions. Whoever wishes, in spite of all difficulties, to pursue the semantics of colloquial language with the help of exact methods will be driven first to undertake the thankless task of a reform of this language. He will find it necessary to define its structure, to overcome the ambiguity of the terms which occur in it, and finally to split the language into a series of languages of greater and greater extent, each of which stands in the same relation to the next in which a formalized language stands to its metalanguage. It may, however, be doubted whether the language of everyday life, after being 'rationalized' in this way, would still preserve its naturalness and whether it would not rather take on the characteristic features of the formalized languages. [Tarski, 1956, p. 267]

However, Donald Davidson had a positive view on the semantical concept of truth with respect to natural language - "the sophisticated and powerful foundation of a competent theory of meaning" [Davidson, 1967, p. 310]. Defending the philosophical and linguistical value of Tarski's theory, he wrote:

> There is no need to suppress, of course, the obvious connection between a definition of truth of the kind Tarski has shown how to construct, and the concept of meaning. It is this: the definition works by giving necessary and sufficient conditions for the truth of every sentence, and to give truth conditions is a way of giving the meaning of a sentence. [Davidson, 1967, p.310]

Davidson argued that most of Tarski's concerns can be eliminated, though he agreed that some problems were still to be overcome. He believed the theory to be the most promising theory for a formal characterization of meaning of natural language.

To introduce his theory Tarski gave an informal but intuitive definition of truth of sentences in natural language:

Definition 1.3. [Truth of a sentence] A **true sentence** is one which says that the state of affairs is so and so, and the state of affairs indeed is so and so.

The definition can be reformulated in a more formal way:

DEFINITION 1.4. [Truth of a sentence] Let s be a sentence. Let t be a function that translates any sentence into a description (in the meta-language) of a state of affairs. Then s is a **true sentence** if and only if $t(s)$.

In other words, to see whether a sentence is true, it has to be **interpreted** and its interpretation has to match the current state of the world. As an illustration of Definition 1.4, consider a few examples below viewing them in the present world (i.e. the world we live in):

(1) the sentence "it is snowing in France" is true if and only if it is the case that in some place in France it is snowing

(2) the sentence "the Eiffel Tower is located on the Champ de Mars in Paris" is true if and only if the Eiffel Tower is located on the Champ de Mars in Paris

(3) "two plus three is equal to five" is a true sentence if and only if two plus three is equal to five

(4) "2+3=5" is a true sentence if and only if two plus three is equal to five

Consider Example (2). For the sentence "the Eiffel Tower is located on the Champ de Mars in Paris" to be true, there should exist three physical objects named "Eiffel Tower", "Champ de Mars" and "Paris". All three objects should be in a particular spatial relation to each other expressed by the words "is located on" and "in". It is possible to continue specifying the entities and the spatial relations formally, but it can be seen already from this informal explanation that what has been described holds in the present world, therefore the sentence "the Eiffel Tower is located on the Champ de Mars in Paris" is true in this world.

It is important to keep in mind the distinction between sentences and descriptions of the state of the affairs. The sentences are expressed with the language under analysis, the **object language**. The descriptions of the state of affairs, which are the statements related to the object language and represent its meanings, are expressed with the **meta-language**. In the examples above English serves both as the object and the meta-language, but it is possible to use any other language (formal or not) as a meta-language. Figure 1.1 shows a general scheme of interpretation of an object language into a meta-language.[1]

It still remains to explain how exactly the description of the state of the affairs required for a sentence to be true (i.e. how the function t is defined) is constructed and what guarantees that the constructed truth conditions match the sentences of the object language. In other words, how absurd interpretations, as for example in (5), are avoided.

[1] In what follows, "language" is sometimes used instead of "object language" when it is clear from the context.

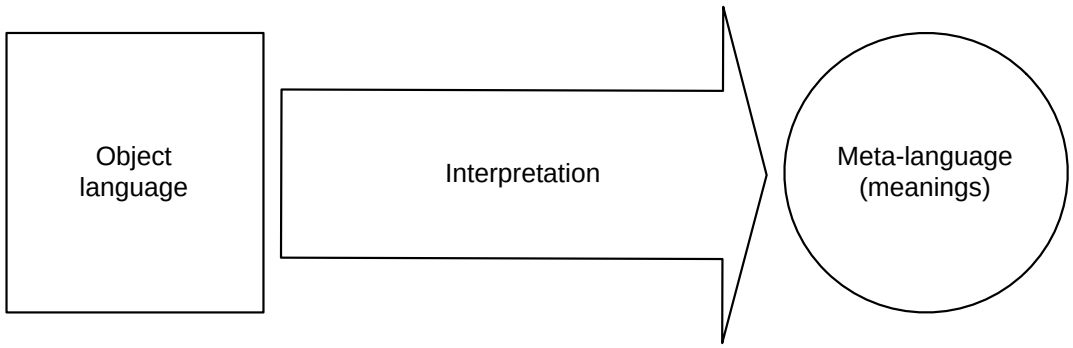


Figure 1.1: Interpretation of an object language.

(5) "it is snowing in France" is a true sentence if and only if the Eiffel Tower is located on the Champ de Mars in Paris

This leads to the notion of "compositionality".

1.2 Compositionality

Sentences of a language are constructed inductively from basic expressions according to particular syntactic rules. For example, sentences in English are constructed from words (if we consider them as primitives) according to the syntax rules of the English language grammar. If we associate each basic expression with what it refers to in a world and if, along with syntactic combination of constituents, we make a parallel semantic combination, we will be able to compute the correct truth conditions of the sentences. This strategy is a fundamental tool in truth-conditional semantics and it is known as the principle of compositionality.[2]

Principle 1.5. [Compositionality] *The meaning of a compound expression is a function of the meanings of its parts and of the syntactic rules by which they are combined.*

Principle 1.5 is rather general and informal. A more formal definition can be given using the algebraic notion of homomorphism, a function between the sets of two algebras respecting the algebraic structure [Grätzer, 1979]:

Definition 1.6. [Algebra] An **algebra** $\langle A, F\rangle$ is a pair, where A is a nonvoid set (the **base set**) and F is a set of operations on A.

[2] See [Dever, 2006] for an extended discussion of compositionality.

DEFINITION 1.7. [Homomorphism] Given two algebras $\langle A, F_A \rangle$ and $\langle B, F_B \rangle$, a mapping $h : A \to B$ is a **homomorphism** from $\langle A, F_A \rangle$ to $\langle B, F_B \rangle$ iff for every n-ary operation $\Theta \in F_A$ there is a n-ary operation $\Psi \in F_B$ such that

$$h(\Theta(a_1, \ldots, a_n)) = \Psi(h(a_1), \ldots, h(a_n))$$

where $a_1, \ldots, a_n$ are elements of A.

When syntactic and semantic operations are viewed as algebraic operations, the interpretation function I should be a homomorphism from syntactic algebra to semantic algebra in order to satisfy the requirement of compositionality.

Figure 1.2 schematically shows the computation of meanings for sentences of a language, where the arrow represents the syntactic processing of the language in parallel to the semantic interpretation with a homomorphic function.

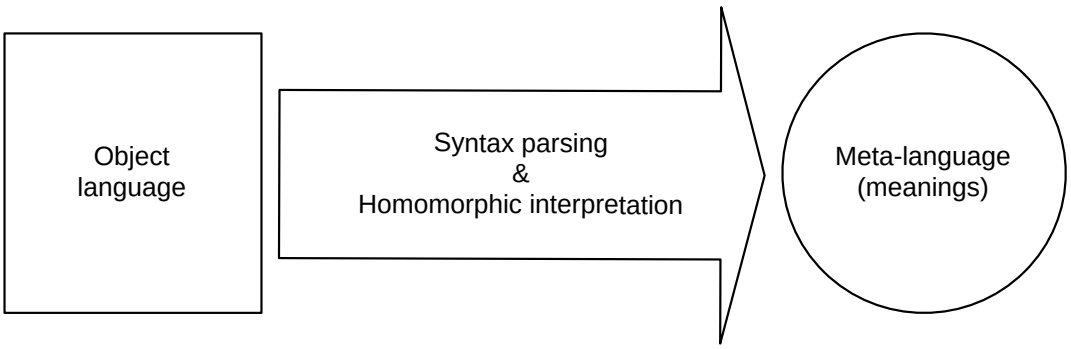


Figure 1.2: Compositional interpretation of a language.

As discussed in Section 1.1, the meaning of a language can be expressed with another (formal or natural) language. It can also be a domain of a model as studied in model-theory (see Section 1.3). The crucial point here is the existence of the homomorphism, which facilitates the translation of sentences of a language into their meanings.

EXAMPLE 1.8. Consider a tiny fragment E of the English language, consisting of two basic expressions, *Alexander* of category NP and *sleeps* of category VP, and one syntactic rule, saying that if t^{NP} and t^{VP} are expressions of category NP and VP respectively, then $\Theta(t^{NP}, t^{VP})$ is an expression of category S, where Θ is the syntactic operation of concatenation, as shown below

$$\Theta(t^{NP}, t^{VP}) \doteq t^{NP}\ t^{VP}$$

With the syntactic rule, the following complex expression can be constructed from the basic expressions in E:

$$\Theta(Alexander, sleeps) = Alexander\ sleeps$$

Consider a fragment F of the French language consisting of two basic expressions, *Alexandre* of category NP' and *dort* of category VP', and one syntactic rule, saying that if $t^{NP'}$ and $t^{VP'}$ are expressions of category NP' and VP' respectively, then $\Psi(t^{NP'}, t^{VP'})$ is an expression of category S', where Ψ is the syntactic operation of concatenation, as shown below

$$\Psi(t^{NP'}, t^{VP'}) \doteq t^{NP'}\ t^{VP'}$$

E is translated into F with the following interpretation function:

$$I(Alexander) = \text{Alexandre}$$
$$I(sleeps) = \text{dort}$$

Then, according to the semantic rule, the following holds:

$$\begin{aligned} I(\Theta(Alexander, sleeps)) &= \Psi(I(Alexander), I(sleeps)) \\ &= \Psi(\text{Alexandre}, \text{dort}) \\ &= \text{Alexandre dort} \end{aligned}$$

The interpretation function I is a homomorphism, therefore compositionality holds. □

It is important to emphasize why compositionality is important. An alternative way of expressing meanings of sentences of a language would be to construct a list of tuples $\langle s, m\rangle$, where s is a sentence and m is the meaning of s. However, since a language consists of infinitely many sentences, the list would be also infinite. Compositionality, in contrast, allows to finitely define an infinite number of meanings of a language. When the principle of compositionality holds, it suffices to define the interpretations of basic expressions because the interpretations of complex expressions can be recursively computed from the basic ones in accordance with syntactic and semantic rules.

Considering that the human brain has finite storage capacity, it is easy to see that the principle of compositionality is also relevant in answering the question how a human being can understand complex expressions without ever hearing them before. Reflections on this topic can already be found at least as early as in writings of Gottlob Frege [Frege, 1914; Frege, 1963]:[3]

> It is astonishing what language can do. With a few syllables it can express an incalculable number of thoughts, so that even a thought grasped by a human being for the very first time can be put into a form of words which will be understood by someone to whom the thought is entirely new. This would be impossible, were we not able to distinguish parts in the thought corresponding to the parts of a sentence, so that the structure of the sentence serves as an image of the structure of the thought. [Frege, 1963, p. 1]

[3]The principle of compositionality should not be confused with the principle of contextuality also presented by Frege. See [Janssen, 1997, p. 420] for discussion.

1.3 Model-Theory

Model-theoretic semantics studies the meanings of expressions of a language by interpreting them as elements in the domains of models.[4]

DEFINITION 1.9. [Model] A **model** $\mathcal{M}$ with respect to a language L is a pair $\langle D, I \rangle$, where D is the **domain** of $\mathcal{M}$ and I is an **interpretation** that maps expressions of L into elements of D.

It is possible to translate Frege's ideas from "Über Sinn und Bedeutung" [1892] to modern model-theoretic terminology. Here are a few pharagraphs from the paper:

> A proper name (word, sign, sign combination, expression) *expresses* its sense, *stands for* or *designates* its reference. By means of a sign we express its sense and designate its reference. [Frege, 1952, p. 61]

> If now the truth value of a sentence is its reference, then on the one hand all true sentences have the same reference and so, on the other hand, do all false sentences. From this we see that in the reference of the sentence all that is specific is obliterated. We can never be concerned only with the reference of a sentence; but again the mere thought alone yields no knowledge, but only the thought together with its reference, i.e. its truth value. Judgments can be regarded as advances from a thought to a truth value. [Frege, 1952, p. 65]

Sense for Frege corresponds to an expression of a language. Sentences represent thoughts. A reference is an element from the domain of some model related to the language. A judgment is an interpretation function mapping the expressions of the language into the elements of the domain. Thoughts (i.e. sentences) are mapped to truth values.

The following examples define different languages and models with respect to these languages.

EXAMPLE 1.10. [Language L] Consider a simple language L, which alphabet contains the following symbols:

1. *constant symbols*: $c_1, \ldots, c_m, \ldots$
2. *predicate symbols*: $P_1, \ldots P_m, \ldots$
3. *function symbols*: $f_1, \ldots, f_m, \ldots$

[4] [Chang and Keisler, 1990] and [Hodges, 1997] are possible readings for getting more mathematically oriented insight into model theory.

Each predicate symbol P_i has an arity, which is a natural number. Each function symbol f_i has an arity, which is a natural number not equal to zero. The expressions of the language are either **terms** or **formulas** (**sentences**) specified by the grammar which is defined inductively as follows:

1. every constant symbol is a term
2. if $t_1, \ldots, t_n$ are terms and if f is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is a term
3. nothing else is a term
4. if $t_1, \ldots, t_n$ are terms and if P is an n-ary predicate symbol, then $P(t_1, \ldots, t_n)$ is a formula
5. nothing else is a formula

□

Example 1.11. [Model w.r.t. L (General)] Let L be the language defined in Example 1.10. A model $\mathcal{M}$ with respect to L is a pair $\langle D_U, I \rangle$, where D_U is the domain and I is an interpretation function that maps the expressions of L into elements of D_U. The domain is defined below, where U is a nonempty set, called the **universe** of $\mathcal{M}$:[5]

$$D_U \doteq \bigcup_{n=0}^{\infty} (U^n \to U) \cup \bigcup_{n=0}^{\infty} U^n$$

The function I is defined as follows:

1. every constant symbol c is mapped into an element $c^I \in U$
2. if $t_1, \ldots, t_n$ are terms and if f is an n-ary function symbol, then f is mapped into n-ary partial function $f^I : U^n \to U$ and the following equation holds:
$$(f(t_1, \ldots, t_n))^I \doteq f^I(t_1^I, \ldots, t_n^I)$$
3. if $t_1, \ldots, t_n$ are terms and if P is an n-ary predicate symbol, then P is mapped into n-ary relation P^I, such that $P^I \subseteq U^n$, and the following equation holds:
$$(P(t_1, \ldots, t_n))^I \doteq (t_1^I, \ldots, t_n^I) \in P^I$$

□

[5] Subsequently, $\top$ is used as an abbreviation for $\{\emptyset\}$ and stands for **true**; and $\bot$ is used as an abbreviation for $\emptyset$ and stands for **false**.

Note that in Examples 1.10 and 1.11 the interpretation I is a function that preserves structure, i.e. it is a **homomorphism**. Thus, the meanings of expressions of the language L from Example 1.10 are computed by interpretation I from Example 1.11 in a compositional manner. Figure 1.3 is a particular case of Figure 1.2. It sketches a model-theoretical compositional interpretation of a language.

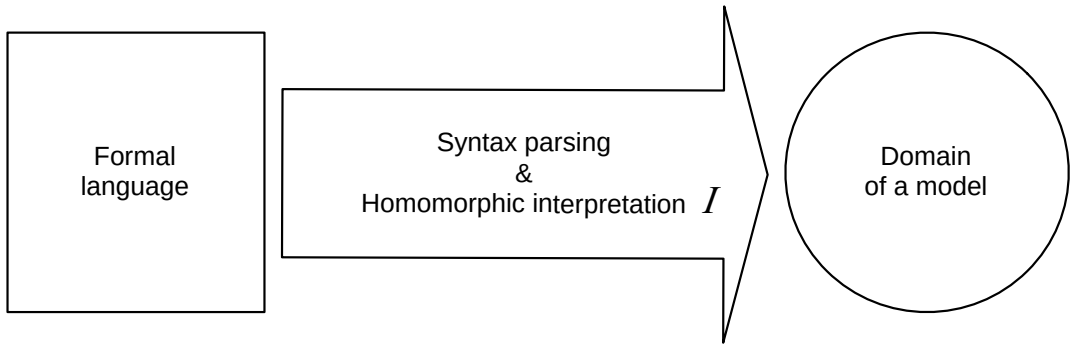


Figure 1.3: Model-theoretical compositional interpretation of a formal language.

For a more intuitive illustration what a model for a language is, Example 1.12 considers a small fragment of the language from Example 1.10. Examples 1.12 and 1.16 define two different models, $\mathcal{M}$ and $\mathcal{M}'$ respectively, of a simple hierarchy of a family members.

EXAMPLE 1.12. [Model for L (Specific)] Let L be the language defined in Example 1.10 with four constants, one predicate and two function symbols. Let model $\mathcal{M}$ with respect to L be a pair $\langle D_U, I\rangle$, where

- $U = \{\text{Mary}, \text{John}, \text{Ann}, \text{Tom}\}$
- I is defined as follows
  1. $c_1^I = \text{Mary}$, $c_2^I = \text{John}$, $c_3^I = \text{Ann}$, $c_4^I = \text{Tom}$
  2. f_1^I is a function that for every element $x \in U$ returns the mother of x:

$$f_1^I(\text{Mary}) = \text{Ann}$$
$$f_1^I(\text{John}) = \text{Ann}$$

  f_2^I is a function that for every element $x \in U$ returns the father of x:

$$f_2^I(\text{Mary}) = \text{Tom}$$
$$f_2^I(\text{John}) = \text{Tom}$$

3. P_1^I is a binary relation between parents:

$$P_1^I = \{(\text{Ann}, \text{Tom}), (\text{Tom}, \text{Ann})\}$$

□

At this point a language and a model related to this language are defined. The language is capable of generating a number of sentences. For any of these sentences, it makes sense to be able to check whether it is true in the defined model.

DEFINITION 1.13. [Truth] Let $\mathcal{M}$ be a model such that $\mathcal{M} = \langle D, I \rangle$. To express the fact that a sentence ξ is **true in** $\mathcal{M}$, the following notation is used: $\mathcal{M} \vDash \xi$, i.e.

$$\mathcal{M} \vDash \xi \quad \text{iff} \quad \xi^I = \top$$

In this case, it is normally said that ξ **holds in** $\mathcal{M}$, or that $\mathcal{M}$ is a **model of** ξ.

EXAMPLE 1.14. Suppose the following string of symbols $P_1(f_2(c_1), f_1(c_2))$ is given, with P_1 being a binary predicate symbol, and f_1 and f_2 being unary function symbols. It can be checked, according to the definitions in Examples 1.10 and 1.11, whether it is a formula that holds in the model $\mathcal{M}$ from Example 1.12:

$$\begin{aligned}(P_1(f_2(c_1), f_1(c_2)))^I &= (f_2^I(c_1^I), f_1^I(c_2^I)) \in P_1^I \\ &= (f_2^I(\text{Mary}), f_1^I(\text{John})) \in P_1^I \\ &= (\text{Tom}, \text{Ann}) \in P_1^I \\ &= \top\end{aligned}$$

Therefore, $\mathcal{M} \vDash P_1(f_2(c_1), f_1(c_2))$. □

However, there may be sentences of a language that are not true in a model:

EXAMPLE 1.15. Suppose another string of symbols $P_1(c_1, c_2)$ is given and it has to be checked whether it is a true formula in the model $\mathcal{M}$ from Example 1.12:

$$\begin{aligned}(P_1(c_1, c_2))^I &= (c_1^I, c_2^I) \in P_1^I \\ &= (\text{Mary}, \text{John}) \in P_1^I \\ &= \bot\end{aligned}$$

Therefore, $P_1(c_1, c_2)$ does not hold in $\mathcal{M}$. □

Examples 1.14 and 1.15 show that, taking one model, there may be some sentences of a language interpreted as true in that model, and other sentences interpreted as false in the model. Moreover, if more than one model is considered, the same sentence may be true in one model, but false in some other model. To illustrate this, the second model $\mathcal{M}'$ with respect to L is defined in Example 1.16, and it is shown in Example 1.17 that the formula $P_1(c_1, c_2)$ holds in $\mathcal{M}'$.

EXAMPLE 1.16. Take the language L defined in Example 1.12. $\mathcal{M}'$ is another model with respect to L that has *the same* universe as $\mathcal{M}$. It is defined as a pair $\langle D'_{U'}, I' \rangle$, where

- $U' = \{\text{Mary}, \text{John}, \text{Ann}, \text{Tom}\}$
- I' is defined as follows
  1. $c_1^{I'} = \text{Mary}$, $c_2^{I'} = \text{John}$, $c_3^{I'} = \text{Ann}$, $c_4^{I'} = \text{Tom}$
  2. $f_1^{I'}$ is a function that for every element $x \in U'$ returns the mother of x:

$$f_1^{I'}(\text{Ann}) = \text{Mary}$$
$$f_1^{I'}(\text{Tom}) = \text{Mary}$$

  $f_2^{I'}$ is a function that for every element $x \in U'$ returns the father of x:

$$f_2^{I'}(\text{Ann}) = \text{John}$$
$$f_2^{I'}(\text{Tom}) = \text{John}$$

  3. $P_1^{I'}$ is a binary relation for couples:

$$P_1^{I'} = \{(\text{Mary}, \text{John}), (\text{John}, \text{Mary})\}$$

□

EXAMPLE 1.17. Recall the string $P_1(c_1, c_2)$ from Example 1.15. This is a true formula in the model $\mathcal{M}'$ from Example 1.16:

$$\begin{aligned}(P_1(c_1, c_2))^{I'} &= (c_1^{I'}, c_2^{I'}) \in P_1^{I'} \\ &= (\text{Mary}, \text{John}) \in P_1^{I'} \\ &= \top\end{aligned}$$

Thus, $\mathcal{M}' \vDash P_1(c_1, c_2)$ □

The sentence $P_1(c_1, c_2)$ of the language L is interpreted in two different models $\mathcal{M}$ and $\mathcal{M}'$ in Examples 1.15 and 1.17. The sentence is interpreted as false in the model $\mathcal{M}$, and as true in the model $\mathcal{M}'$. These examples demonstrate that, from a model-theoretical perspective, it only makes sense to talk about truth in a model. In other words, if we have a sentence in some language, we first have to define a model, and only afterwards we can raise the question whether the sentence is true or false in the defined model by checking how the sentence is interpreted in the model.

The relativity of truth holds for any language, even for a natural language. Frege observed it in "Über Sinn und Bedeutung" [1892], where he says:

> Let us assume for the time being that the sentence has reference. If we now replace one word of the sentence by another having the same reference, but a different sense, this can have no bearing upon the reference of the sentence. Yet we can see that in such a case the thought changes; since, e.g., the thought in the sentence 'The morning star is a body illuminated by the Sun' differs from that in the sentence 'The evening star is a body illuminated by the Sun.' Anybody who did not know that the evening star is the morning star might hold the one thought to be true, the other false. The thought, accordingly, cannot be the reference of the sentence, but must rather be considered as the sense. [Frege, 1952, p. 62]

In the quoted paragraph Frege shows an example that a sentence can have one interpretation (e.g. mapped to true) in one case, and another one (false) in some other case. The example is dealing, in fact, with two different models. The first model is a model which interpretation maps 'the morning star' and 'the evening star' into the same element of its domain. In this model, the sentences 'The morning star is a body illuminated by the Sun' and 'The evening star is a body illuminated by the Sun' are mapped to the (same) truth value true (i.e. $\top$). The second model is for those who were not aware that the evening star is the morning star: 'the morning star' and 'the evening star' are mapped into different elements of the domain of the model. Then, indeed, in the second model one of the sentences can be mapped into the truth value true, while the other to false.

In the next two examples the relativity of truth is illustrated in detail. A fragment of the English language is considered and the sentence 'the morning star is the evening star' is interpreted in two different models.

Example 1.18. Consider a tiny fragment of primitive English consisting of two noun phrases: $TheMorningStar$ and $TheEveningStar$; and one verb: is. The grammar of the language consists of a single rule, saying that a sentence is defined as a noun phrase followed by a verb followed by a noun phrase. Consider the following sentence in the language just defined:

(6) $TheMorningStar\ is\ TheEveningStar$

Take the model $\mathcal{M}$, such that $\mathcal{M} = \langle D_U, I\rangle$, where

- $U = \{\text{Venus}, \text{Mars}, \text{Jupiter}, \text{Earth}\}$
- the interpretation I is defined as follows

$$TheMorningStar^I = \text{Venus}$$
$$TheEveningStar^I = \text{Venus}$$
$$is^I = \{(x, x) | x \in D\}$$

i.e. $is^I = \{(\text{Venus}, \text{Venus}), (\text{Mars}, \text{Mars}), (\text{Jupiter}, \text{Jupiter}), (\text{Earth}, \text{Earth})\}$
For every noun phrases n_1 and n_2, and for every verb v the following equation holds:

$$(n_1\ v\ n_2)^I \doteq (n_1^I, n_2^I) \in v^I$$

It can be checked now whether the sentence (6) is true or false in the model $\mathcal{M}$:

$$\begin{aligned}(&TheMorningStar\ is\ TheEveningStar)^I \\ &= (TheMorningStar^I, TheEveningStar^I) \in is^I \\ &= (\text{Venus}, \text{Venus}) \in is^I \\ &= \top\end{aligned}$$

□

Example 1.19. Imagine a situation in which the humanity lives in an alternative world where the morning star and the evening star are not the same planet. A model $\mathcal{M}'$ for this situation with respect to the language can be defined, as shown in Example 1.18. $\mathcal{M}'$ is defined as a tuple $\langle D'_{U'}, I' \rangle$, where

- $U' = \{\text{Phosphorus}, \text{Hesperus}, \text{Mars}, \text{Jupiter}, \text{Earth}\}$
- the interpretation I' is defined as follows

$$\begin{aligned}TheMorningStar^{I'} &= \text{Phosphorus} \\ TheEveningStar^{I'} &= \text{Hesperus} \\ is^{I'} &= \{(x, x) | x \in D'\}\end{aligned}$$

such that for every noun phrases n_1 and n_2, and for every verb v the following equation holds:

$$(n_1\ v\ n_2)^{I'} \doteq (n_1^{I'}, n_2^{I'}) \in v^{I'}$$

We can now check whether the sentence (6) holds in the model $\mathcal{M}'$:

$$\begin{aligned}(&TheMorningStar\ is\ TheEveningStar)^{I'} \\ &= (TheMorningStar^{I'}, TheEveningStar^{I'}) \in is^{I'} \\ &= (\text{Phosphorus}, \text{Hesperus}) \in is^{I'} \\ &= \bot\end{aligned}$$

□

As the previous examples show, models can be different. Whatever expression is generated by a language, it may have different interpretations in different models. The models presented so far were close to our awareness of the present world (the world we live in), but it does not have to be so, it can perfectly be a model of any alternative world. A domain can consist of any elements, and an interpretation function can be defined in all possible ways to map the expressions of the language to the elements of the domain. This generality of model theory allows to have a simple treatment of the long philosophical debates regarding the meanings of sentences like "The king of France is bald". Example (1.20) defines three models and checks the interpretation of the sentence in them.

Example 1.20. Consider the language having the following basic constituents: two nouns *France* and *Belgium*, a verb *is*, an adjective *bald*, a functional phrase *the king of*. The grammar of the language contains a single rule saying that a sentence can be formed according to the pattern $f\ n\ v\ a$, where f stands for the functional phrase, n stands for a noun, v stands for the verb, a stands for the adjective.

Consider the following sentence in the language:

(7) *the king of France is bald*

Below this sentence is interpreted in three models $\langle D_U, I_i\rangle$ for $i \in \{1,2,3\}$. All three models share the same domain D_U with the universe U defined as:

$$U = \{\text{Albert II, John, Tom, France, Belgium}\}$$

For all three interpretation functions I_i, $i \in \{1,2,3\}$, the following equations, where f is the functional phrase, n is a noun, v is the verb and a is the adjective, hold:

$$(f\ n)^{I_i} \doteq f^{I_i}(n^{I_i})$$
$$(f\ n\ v\ a)^{I_i} \doteq ((f\ n)^{I_i}, a^{I_i}) \in\ v^{I_i}$$

Model 1: The model $\mathcal{M}_1$ is the model $\langle D_U, I_1\rangle$, where I_1 is defined as follows:

$$\begin{aligned} France^{I_1} &= \text{France} \\ Belgium^{I_1} &= \text{Belgium} \\ bald^{I_1} &= \{\text{Albert II, John}\} \\ is^{I_1} &= \{(x,Y) | x \in Y, Y \in 2^U\} \end{aligned}$$

The functional phrase is interpreted as a function that, when given an element from the domain which is a country, returns the king of this country. The function

I_1 defines that in the model $\mathcal{M}_1$ the king of France is John and the king of Belgium is Albert II.

$$\begin{aligned}(\textit{the king of})^{I_1}(\text{France}) &= \text{John}\\ (\textit{the king of})^{I_1}(\text{Belgium}) &= \text{Albert II}\end{aligned}$$

We can now interpret sentence (7) in the model $\mathcal{M}_1$:

$$\begin{aligned}&(\textit{the king of France is bald})^{I_1}\\ &\quad= ((\textit{the king of France})^{I_1}, \textit{bald}^{I_1}) \in \textit{is}^{I_1}\\ &\quad= ((\textit{the king of})^{I_1}\textit{France}^{I_1}, \textit{bald}^{I_1}) \in \textit{is}^{I_1}\\ &\quad= ((\textit{the king of})^{I_1}\text{France}, \textit{bald}^{I_1}) \in \textit{is}^{I_1}\\ &\quad= (\text{John}, \textit{bald}^{I_1}) \in \textit{is}^{I_1}\\ &\quad= (\text{John}, \{\text{Albert II, John}\}) \in \textit{is}^{I_1}\\ &\quad= (\text{John}, \{\text{Albert II, John}\}) \in \{(\text{John}, \{\text{Albert II, John}\}),\\ &\qquad\qquad\qquad\qquad (\text{Albert II}, \{\text{Albert II, John}\}),\}\\ &\qquad\qquad\qquad\qquad (\text{John}, \{\text{John, Belgium}\}),\}\\ &\qquad\qquad\qquad\qquad (\text{Albert II}, \{\text{John, Belgium}\}),\}\\ &\qquad\qquad\qquad\qquad \ldots\}\\ &\quad= \top\end{aligned}$$

The sentence is interpreted as true in the model $\mathcal{M}_1$ because the element (John) of the universe returned by the interpretation of the functional phrase applied to France (which is also an element of the domain) is in the set of elements defined by the interpretation of the adjective *bald* (i.e. the set of bald people in the model).

Model 2: The second model $\mathcal{M}_2$ is defined as a tuple $\langle D_U, I_2\rangle$, where the interpretation I_2 is as follows:

$$\begin{aligned}\textit{France}^{I_2} &= \text{France}\\ \textit{Belgium}^{I_2} &= \text{Belgium}\\ \textit{bald}^{I_2} &= \{\text{Albert II, Tom}\}\\ \textit{is}^{I_2} &= \{(x, Y) | x \in Y, Y \in 2^U\}\\ (\textit{the king of})^{I_2}(\text{France}) &= \text{John}\\ (\textit{the king of})^{I_2}(\text{Belgium}) &= \text{Albert II}\end{aligned}$$

Then, the sentence (7) is interpreted as false in the model $\mathcal{M}_2$:

$$
\begin{aligned}
&(\textit{the king of France is bald})^{I_2} \\
&\quad = ((\textit{the king of France})^{I_2}, \textit{bald}^{I_2}) \in \textit{is}^{I_2} \\
&\quad = ((\textit{the king of})^{I_2}\textit{France}^{I_2}, \textit{bald}^{I_2}) \in \textit{is}^{I_2} \\
&\quad = ((\textit{the king of})^{I_2}\text{France}, \textit{bald}^{I_2}) \in \textit{is}^{I_2} \\
&\quad = (\text{John}, \textit{bald}^{I_2}) \in \textit{is}^{I_2} \\
&\quad = (\text{John}, \{\text{Albert II, Tom}\}) \in \textit{is}^{I_2} \\
&\quad = (\text{John}, \{\text{Albert II, Tom}\}) \in \{(\text{John}, \{\text{Albert II, Tom}\}), \\
&\qquad\qquad\qquad\qquad (\text{Albert II}, \{\text{Albert II, Tom}\}), \\
&\qquad\qquad\qquad\qquad (\text{John}, \{\text{John, Belgium}\}), \} \\
&\qquad\qquad\qquad\qquad (\text{Albert II}, \{\text{John, Belgium}\}), \} \\
&\qquad\qquad\qquad\qquad \ldots\} \\
&\quad = \bot
\end{aligned}
$$

Note that the sentence is interpreted as false because the individual John returned by the interpretation of the functional phrase when applied to France (i.e. the king of France in the model) is not in the set of bald people.

Model 3: The interpretation I_3 of the third model $\mathcal{M}_3$ equal to $\langle D_U, I_3 \rangle$ is defined in the following way:

$$
\begin{aligned}
\textit{France}^{I_3} &= \text{France} \\
\textit{Belgium}^{I_3} &= \text{Belgium} \\
\textit{bald}^{I_3} &= \{\text{Albert II, Tom}\} \\
\textit{is}^{I_3} &= \{(x, Y) | x \in Y, Y \in 2^U\}
\end{aligned}
$$

By relaxing the notion of model and allowing the interpretation function I_3 to be partial, the functional phrase can be defined by I_3 only for Belgium:

$$(\textit{the king of})^{I_3}(\text{Belgium}) = \text{Albert II}$$

Since the interpretation function is undefined for France, sentence (7) cannot be interpreted in the model $\mathcal{M}_3$:

$$
\begin{aligned}
&(\textit{the king of France is bald})^{I_3} \\
&\quad = ((\textit{the king of France})^{I_3}, \textit{bald}^{I_3}) \in \textit{is}^{I_3} \\
&\quad = ((\textit{the king of})^{I_3}\textit{France}^{I_3}, \textit{bald}^{I_3}) \in \textit{is}^{I_3} \\
&\quad = ((\textit{the king of})^{I_3}\text{France}, \textit{bald}^{I_3}) \in \textit{is}^{I_3} \\
&\quad = ???
\end{aligned}
$$

Thus, sentence (7) is simply meaningless in the model $\mathcal{M}_3$. □

Example 1.20 shows that the same sentence may not only be interpreted as true or false, but also fail to have an interpretation in a model, therefore be meaningless. This raises at least two questions: first, if there exist meaningful sentences that are always true (i.e. true in all models), and second, in the case there is some knowledge base consisting of a set of sentences, which meaningful sentences are able to contribute knowledge to this knowledge base.

Definition 1.21. [Valid sentence] A sentence ξ of a language L is **valid** if and only if it is true in all models of L. Formally:

$$\vDash \xi \quad \text{iff} \quad \mathcal{M} \vDash \xi \text{ for all } \mathcal{M}$$

Definition 1.22. [Entailment] Let $\xi, \xi_1, \ldots, \xi_n$ be sentences in a language L. A set of sentences $\{\xi_1, \ldots, \xi_n\}$ **entails** the sentence ξ if and only if in all models in which all the sentences $\xi_1, \ldots, \xi_n$ are true, the sentence ξ is also true. Or equivalently and more formally:

$$\{\xi_1, \ldots, \xi_n\} \vDash \xi \quad \text{iff} \quad \text{for all } \mathcal{M}, \text{ if for all } \xi_i \ \ \mathcal{M} \vDash \xi_i \text{ then } \mathcal{M} \vDash \xi$$

Imagine we have a knowledge base consisting of a set of sentences and consider the class of all models in which all the sentences of the knowledge base are true. Then, sentences that are valid in this class of models are **not informative**. **Informative sentences** are those that are true in some of the models, but not valid.

The model-theoretic view of "informativeness" of a sentence is compatible with Frege's [1892] theory. One of the examples that Frege has considered included a discussion of the cognitive values (i.e informativeness) of the sentences $a{=}a$ and $a{=}b$ of some language. Here is what Frege says concluding his paper:

> When we found '$a{=}a$' and '$a{=}b$' to have different cognitive values, the explanation is that for the purpose of knowledge, the sense of the sentence, viz., the thought expressed by it, is no less relevant than its reference, i.e. its truth value. If now $a{=}b$, then indeed the reference of 'b' is the same as that of 'a', and hence the truth value of '$a{=}b$' is the same as that of '$a{=}a$'. In spite of this, the sense of 'b' may differ from that of 'a', and thereby the thought expressed in '$a{=}b$' differs from that of '$a{=}a$'. In that case the two sentences do not have the same cognitive value. If we understand by 'judgement' the advance from the thought to its truth value, [...] we can also say that the judgements are different. [Frege, 1952, p. 78]

By giving a clear distinction between sense (an expression) and reference (an element of a domain), and by considering the possibility of existence of different judgements (interpretations), Frege came very close to the explanation why sentences like $a{=}a$ and $a{=}b$ have different cognitive value (assuming the usual interpretation for equality): the sentence $a{=}a$ is valid, therefore is not informative, whereas the sentence $a{=}b$ can be true in some (but not all) models, therefore it is informative.

1.4 Montague Semantics

Previous sections illustrated that the model-theoretical approach to semantics is very promising. However, model-theoretical interpretations of the simple languages in Section 1.3 were only possible due to the fact that those languages were given a precisely defined structure. In contrast, natural languages are clearly more complex and formalization of natural language meaning requires a very careful definition of grammar for fragments of natural languages. This has already been observed by Tarski:

> *The problem of the definition of truth obtains a precise meaning and can be solved in a rigorous way only for those languages whose structure has been exactly specified.* For other languages - thus, for all natural, "spoken" languages - the meaning of the problem is more or less vague, and its solution can have only an approximate character. Roughly speaking, the approximation consists in replacing a natural language (or a portion of it in which we are interested) by one whose structure is exactly specified, and which diverges from the given language "as little as possible." [Tarski, 1944]

The strategy of analysing the fragments of natural language together with the compositionality principle (empowering the computation of meanings by implementing semantic rules of a grammar in parallel with syntactic rules) allowed to characterize natural language semantics in a concise systematic way with formal mathematical tools. Richard Montague is considered [Partee, 1975] to be the first researcher to do it:

> There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory. [Montague, 1970b, p.373]

In [1970b] Montague presented his general theory of universal syntax and semantics, and in the papers [1970a] and [1973] he gave grammars for limited

fragments of English.[6]

The fact that Montague used standard tools from mathematical logic to assign semantics to natural language utterances results in (at least) two fundamental characteristics of his theory. First of all, Montague's semantic is compositional as defined in Section 1.2. Therefore, for Montague the function of syntax is not only to generate well-formed expressions, but also to provide the structural basis for their semantic interpretation. Another characteristic of Montague semantics is that it employs model theory for the interpretation of a language. Montague gave model-theoretic interpretation to a (English) language in two ways, called "direct" and "indirect" interpretations [Dowty *et al.*, 1981; Partee, 1997b]. The direct interpretation presented in [Montague, 1970a] follows the scheme described in Section 1.3. The indirect interpretation, presented in [Montague, 1970b; Montague, 1973], involves an intermediate language expressed in **intensional logic**. It first translates each expression of an object language into a logical expression of intensional logic, and then this logical expression is model-theoretically interpreted. Figure 1.4 schematically illustrates Montague's indirect interpretation.

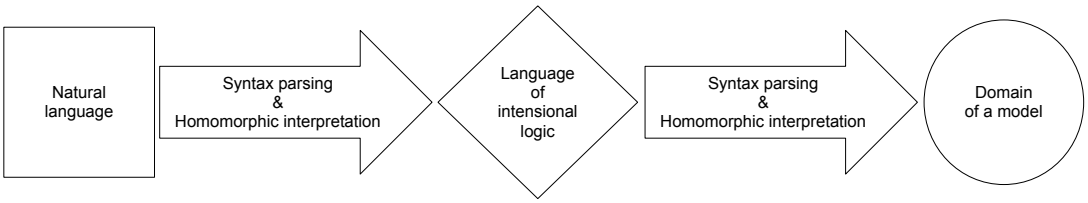


Figure 1.4: Montague's "indirect" model-theoretic interpretation of a language.

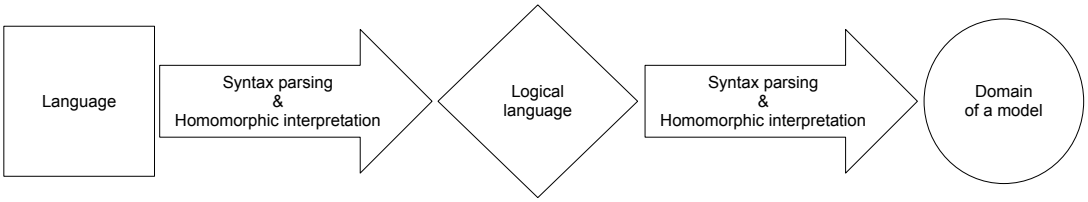


Figure 1.5: "Indirect" model-theoretic interpretation of a language.

Both interpretations should be homomorphisms to guarantee the compositionality principle for the whole indirect interpretation. The following theorem

[6] All three papers are formal and require good mathematical background. Detailed introduction into Montague semantics can be found in [Partee, 1975]. [Dowty *et al.*, 1981] explains Montague semantics in a systematic way, starting from a simple formal language and concluding with explicit explanation of Montague's intensional logic. The historical context of emergence of Montague semantics can be found in [Partee, 1997a] and [Partee, 1997b].

proves that the composition of homomorphisms indeed preserves the compositionality principle.

THEOREM 1.23. [Composition of homomorphisms] *Let A, B and C be algebraic structures. Let h_1 be a homomorphism from A to B and h_2 be a homomorphism from B to C. Then the composition $h_1 \circ h_2$, defined by $(h_1 \circ h_2)(a) = h_2(h_1(a))$, is a homomorphism from A to C.*

Proof. Let Θ, Ψ and Υ be operations on A, B and C correspondingly, such that $h_1(\Theta) = \Psi$ and $h_2(\Psi) = \Upsilon$. Then

$$\begin{aligned}(h_1 \circ h_2)(\Theta(a_1, \ldots, a_n)) &= h_2(h_1(\Theta(a_1, \ldots, a_n))) \\ &= h_2(\Psi(h_1(a_1), \ldots, h_1(a_n))) \\ &= \Upsilon(h_2(h_1(a_1)), \ldots, (h_2(h_1(a_n)))) \\ &= \Upsilon((h_1 \circ h_2)(a_1), \ldots, (h_1 \circ h_2)(a_n))\end{aligned}$$

□

A trivial collorary of Theorem 1.23 is that any number of intermediate languages can be involved in the compositional interpretation of a language as long as intermediate interpretations are homomorphisms. Moreover, the intermediate language can be eliminated since the composition of homomorphisms is equivalent to the direct interpretation of a language.

Although the intermediate formal language is sometimes neglected in model-theoretic interpretation of a natural language [Jacobson, 1997], there are many advantages in having an indirect interpretation.

While the problem of determining whether a natural language sentence is true or false in a model is usually controversial because of inherent complexity of natural language, for formal languages it is a well understood problem in model-theory. Therefore, after interpreting a natural language into a formal language, we can rely on model-theory to interpret the sentences of the formal language with respect to a model.

Another complex problem in natural language semantics is whether a sentence entails another sentence. However, since the notion of entailment is already well-defined for sentences of formal languages, and logical calculi together with proof search procedures for these formal languages exist, one can rely on these existing techniques if a natural language is first interpreted into a formal language.

Taking these observations into consideration, in this dissertation **formal semantics of a (natural) language** is understood as the procedure of interpreting a (natural) language into a logical language.

To present his theory, Montague defines in [1973] a particular syntax of a fragment of English consisting of basic rules that assign a syntactic category

to each primitive element (i.e. lexical item) and recursive rules that construct expressions.

In his definition of syntax, Montague uses some notions of categorial grammar [Ajdukiewicz, 1935] and his semantics is based on higher-order logic and on the simply-typed lambda calculus.[7] Thus, the foundations of both syntax and semantics in Montague's approach are type-theoretic[8], and, moreover, the syntax-semantics correspondence was later shown in [van Benthem, 1986; van Benthem, 1988] to go in the lines of Curry-Howard isomorphism. Consider the following tiny type-logical grammar which can be used for syntactically forming Sentence (8):

$$\begin{aligned} \mathit{Mary} &: np \\ \mathit{John} &: np \\ \mathit{loves} &: np \to np \to s \end{aligned}$$

(8) *John loves Mary*

The parse structure of Sentence (8) is a proof of the following sequent: $np, np \to np \to s, np \vdash s$, as shown below:[9]

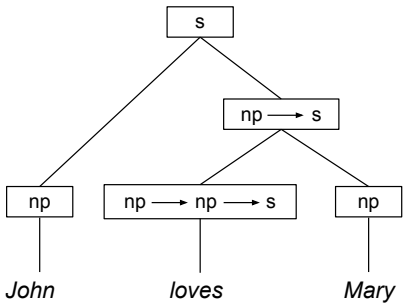


Figure 1.6: Syntactic parse tree of sentence *John loves Mary*.

$$\dfrac{np \vdash np \qquad \dfrac{np \to np \to s \vdash np \to np \to s \qquad np \vdash np}{np \to np \to s, np \vdash np \to s}}{np, np \to np \to s, np \vdash s}$$

[7] Basics of simply-typed lambda calculus are given in Appendix A.

[8] The theory of types is a logical theory, that divides entities into a hierarchy of types. It was introduced by Bertrand Russell as a solution for some contradictions, such as Russell's paradox, in set theory [Russell, 1903]. The creation of a hierarchy of types and the assignment of each entity to a type, such that complex objects are built up from objects of preceding types, guarantees that the type of a property is never the same as the type of entities to which it applies. Therefore, a property never applies to itself.

[9] Often a syntactic parse tree is used to represent the parse structure of the sentence. Figure 1.6 shows the parse tree for Sentence (8).

By the Curry-Howard isomorphism, this proof can be interpreted as the "intermediary" λ-term $(loves\ Mary)\ John$:

$$\dfrac{John : np \vdash John : np \qquad \dfrac{loves : np \to np \to s \vdash loves : np \to np \to s \qquad Mary : np \vdash Mary : np}{loves : np \to np \to s, Mary : np \vdash loves\ Mary : np \to s}}{John : np, loves : np \to np \to s, Mary : np \vdash (loves\ Mary)\ John : s}$$

This λ-term is used to compositionally compute the logical (semantic) form of the sentence from the lexical interpretations. It is "intermediary" because each of the words that compose this term still have to be interpreted.

The evidence of this correspondence between syntax and semantics played an important role in recognition of type-logical grammars originated in [Lambek, 1958; Lambek, 1961] as logics for reasoning about the composition of syntactic forms and hence for obtaining meanings of natural language expressions. Type-logical grammars allow to define natural language syntax and to represent Montague's compositionality program in a more clear, concise and systematic way than when using his original definition of syntax. Therefore, some details of Montague's framework are presented below using type-logical categories and grammars.[10]

There are two particularly notable aspects of Montague's semantics [1973]: his interpretation of (quantified) noun phrases, and his treatment of scope ambiguities.

Under the simplest assignment of types to categories, the category of noun phrases is interpreted as the type of an individual ι and the category of sentences is interpreted as the type of a proposition o.[11] Then the proof above can be interpreted as follows:

$$\dfrac{[\![John]\!] : \iota \vdash [\![John]\!] : \iota \qquad \dfrac{[\![loves]\!] : \iota \to \iota \to o \vdash [\![loves]\!] : \iota \to \iota \to o \qquad [\![Mary]\!] : \iota \vdash [\![Mary]\!] : \iota}{[\![loves]\!] : \iota \to \iota \to o, [\![Mary]\!] : \iota \vdash [\![loves]\!]\ [\![Mary]\!] : \iota \to o}}{[\![John]\!] : \iota, [\![loves]\!] : \iota \to \iota \to o, [\![Mary]\!] : \iota \vdash ([\![loves]\!]\ [\![Mary]\!])\ [\![John]\!] : o}$$

In this case, lexical interpretations for *John*, *Mary* and *loves* can be represented with the following λ-terms, where $\mathbf{j}$ and $\mathbf{m}$ are constants of type ι and $\mathbf{love}$ is a constant of type $(\iota \to \iota \to o)$:

$$[\![Mary]\!] = \mathbf{m} \tag{1.1}$$

$$[\![John]\!] = \mathbf{j} \tag{1.2}$$

$$[\![loves]\!] = \lambda xy.\mathbf{love}\ y\ x \tag{1.3}$$

Then, the meaning of the sentence (8) can be compositionally computed, resulting in (1.4):

$$\begin{aligned} [\![loves]\!][\![Mary]\!][\![John]\!] &= (\lambda xy.\mathbf{love}yx)\ \mathbf{m}\ \mathbf{j} \\ &\to^*_\beta \mathbf{love}\ \mathbf{j}\ \mathbf{m} \end{aligned} \tag{1.4}$$

[10] The rest of the thesis also uses categorial grammar for syntactic parsing of natural language.

[11] For simplicity, Montague's semantics is sketched here ignoring his account of intensionality.

However, quantified noun phrases, such as *everybody*, *somebody* and *nobody*, cannot be interpreted as constants, but as more complex terms shown below in (1.5), and therefore their meanings are of a more complex type $((\iota \to o) \to o)$.

$$\begin{aligned} [\![everybody]\!] &= \lambda P.\forall x.Px \\ [\![somebody]\!] &= \lambda P.\exists x.Px \\ [\![nobody]\!] &= \lambda P.\neg\exists x.Px \end{aligned} \tag{1.5}$$

In set-theoretic terms, the type $(\iota \to o)$ can be seen as a set of individuals (for which a property holds). Then, the type $((\iota \to o) \to o)$ is a set of properties of individuals (i.e. set of sets of individuals).

Montague's innovation was to say that proper names can also be interpreted as terms of type $((\iota \to o) \to o)$, i.e. they can be **type-raised**. Then, for example, the interpretation of *John* is the set of all properties of the individual John[12] and the corresponding λ-term is shown in (1.6):[13]

$$[\![John]\!] = \lambda P.P\mathbf{j} \tag{1.6}$$

Type raised interpretation of noun phrases allowed Montague to handle scope ambiguities in sentences like (9):

(9) a. *John seeks a unicorn.*

b. *Everybody loves somebody.*

The idea is that the new, type raised, interpretations of noun phrases give more flexibility with respect to the functional application, thus allowing semantic content of a subject to appear, after β-reduction, either before the semantic content of an object, leading to subject wide scope reading, or after it, leading to an object wide scope reading. The different orders of functional applications are based on different syntactic trees. Montague's treatment of Sentence (9a) can be found in [Montague, 1973], and [Partee, 1975; Dowty *et al.*, 1981] explain it in detail.

However, modern approaches to semantics go a step further and type-raise verbs as well. Therefore, transitive verbs are interpreted as λ-terms of type $(((\iota \to o) \to o) \to ((\iota \to o) \to o) \to o)$. This makes it possible to handle scope ambiguities not by using different syntactic trees (hence different order of functional applications of lexical items), but by making the lexical interpretation of the transitive verb responsible for placing the lexical contents of the subject

[12] According to [Partee, 1997b], Montague was probably inspired by the ideas presented in [Lewis, 1970].

[13] This kind of transformation of the term (1.2) of type ι into the term (1.6) of type $((\iota \to o) \to o)$ is analogous to a transformation of the term into an equivalent term with a continuation. Chapter 3 introduces the method of continuations and discusses (recent) successful applications of this technique in natural language semantics.

and the object in the appropriate place of the logical formula depending on the reading. The subject wide scope and the object wide scope lexical interpretations of the verb *loves*, for example, are shown in (1.7a) and (1.7b) respectively:

$$[\![loves]\!]_{sws} = \lambda OS.S(\lambda x.O(\lambda y.\mathbf{love}xy)) \tag{1.7a}$$
$$[\![loves]\!]_{ows} = \lambda OS.O(\lambda y.S(\lambda x.\mathbf{love}xy)) \tag{1.7b}$$

When the subject or the object of a transitive verb is a proper name, subject wide scope and object wide scope interpretations of the verb result in equal interpretations for the sentence. This can be seen for the sentence *John loves Mary*, interpretations (1.8) and (1.9) of which are compositionally computed using different interpretations of the verb:

$$\begin{aligned}
&[\![loves]\!]_{sws}[\![Mary]\!][\![John]\!] \\
&= (\lambda OS.S(\lambda x.O(\lambda y.\mathbf{love}xy)))[\![Mary]\!][\![John]\!] \\
&\rightarrow^*_\beta [\![John]\!](\lambda x.[\![Mary]\!](\lambda y.\mathbf{love}xy)) \\
&= [\![John]\!](\lambda x.(\lambda P.P\mathbf{m})(\lambda y.\mathbf{love}xy)) \\
&\rightarrow_\beta [\![John]\!](\lambda x.(\lambda y.\mathbf{love}xy)\mathbf{m}) \\
&\rightarrow_\beta [\![John]\!](\lambda x.\mathbf{love}\ x\ \mathbf{m}) \\
&= (\lambda P.P\mathbf{j})(\lambda x.\mathbf{love}\ x\ \mathbf{m}) \\
&\rightarrow_\beta (\lambda x.\mathbf{love}\ x\ \mathbf{m})\ \mathbf{j} \\
&\rightarrow_\beta \mathbf{love\ j\ m}
\end{aligned} \tag{1.8}$$

$$\begin{aligned}
&[\![loves]\!]_{ows}[\![Mary]\!][\![John]\!] \\
&= (\lambda OS.O(\lambda y.S(\lambda x.\mathbf{love}xy)))[\![Mary]\!][\![John]\!] \\
&\rightarrow^*_\beta [\![Mary]\!](\lambda y.[\![John]\!](\lambda x.\mathbf{love}xy)) \\
&= [\![Mary]\!](\lambda y.(\lambda P.P\mathbf{j})(\lambda x.\mathbf{love}xy)) \\
&\rightarrow_\beta [\![Mary]\!](\lambda y.(\lambda x.\mathbf{love}xy)\mathbf{j}) \\
&\rightarrow_\beta [\![Mary]\!](\lambda y.\mathbf{lovej}y) \\
&= (\lambda P.P\mathbf{m})(\lambda y.\mathbf{lovej}y) \\
&= (\lambda y.\mathbf{lovej}y)\mathbf{m} \\
&\rightarrow_\beta \mathbf{love\ j\ m}
\end{aligned} \tag{1.9}$$

However, sentences with scope ambiguities, like (9b), get different interpretations depending whether subject wide scope or object wide scope lexical interpre-

tations of the verb are used, as shown in (1.10) and (1.11) respectively:

$$
\begin{aligned}
& [\![loves]\!]_{sws}[\![somebody]\!][\![everybody]\!] \\
=\ & (\lambda OS.S(\lambda x.O(\lambda y.\mathbf{love}xy)))[\![somebody]\!][\![everybody]\!] \\
\rightarrow_\beta^*\ & [\![everybody]\!](\lambda x.[\![somebody]\!](\lambda y.\mathbf{love}xy)) \\
=\ & [\![everybody]\!](\lambda x.(\lambda P.\exists z.Pz)(\lambda y.\mathbf{love}xy)) \\
\rightarrow_\beta\ & [\![everybody]\!](\lambda x.\exists z.(\lambda y.\mathbf{love}xy)z) \\
\rightarrow_\beta\ & [\![everybody]\!](\lambda x.\exists z.\mathbf{love}xz) \\
=\ & (\lambda P.\forall y.Py)(\lambda x.\exists z.\mathbf{love}xz) \\
\rightarrow_\beta\ & \forall y.(\lambda x.\exists z.\mathbf{love}xz)y \\
\rightarrow_\beta\ & \forall y.\exists z.\mathbf{love}yz
\end{aligned}
\tag{1.10}
$$

$$
\begin{aligned}
& [\![loves]\!]_{ows}[\![somebody]\!][\![everybody]\!] \\
=\ & (\lambda OS.O(\lambda y.S(\lambda x.\mathbf{love}xy)))[\![somebody]\!][\![everybody]\!] \\
\rightarrow_\beta^*\ & [\![somebody]\!](\lambda y.[\![everybody]\!](\lambda x.\mathbf{love}xy)) \\
=\ & [\![somebody]\!](\lambda y.(\lambda P.\forall z.Pz)(\lambda x.\mathbf{love}xy)) \\
\rightarrow_\beta\ & [\![somebody]\!](\lambda y.\forall z.(\lambda x.\mathbf{love}xy)z) \\
\rightarrow_\beta\ & [\![somebody]\!](\lambda y.\forall z.\mathbf{love}zy) \\
=\ & (\lambda P.\exists x.Px)(\lambda y.\forall z.\mathbf{love}zy) \\
\rightarrow_\beta\ & \exists x.(\lambda y.\forall z.\mathbf{love}zy)x \\
\rightarrow_\beta\ & \exists x.\forall z.\mathbf{love}zx
\end{aligned}
\tag{1.11}
$$

Montague's "English as a formal language" theory lead to a substantial progress in the area of natural language semantics. As mentioned above, it motivated the use of type-logical grammars and type theory providing powerful tools to study derivational and lexical semantics. It, moreover, stimulated the recognition of model theoretical semantics. Montague showed how the compositionality principle can be incorporated into natural language semantics in a mathematically precise way. His work evolved and was continued by many other researches in areas of syntax, semantics and pragmatics of natural language.[14]

1.5 Abstract Categorial Grammars

Abstract categorial grammar [de Groote, 2001a] is an expressive categorial formalism inspired by Montague's ideas and his formalization of syntax-semantics

[14]Many references can be found in [Partee, 1997a]. Among more recent ones are the developments in Lambek-Grishin calculus [Moortgat, 1997; Bernardi and Moortgat, 2010], new grammatical formalisms such as lambda-grammars [Muskens, 2001] and abstract categorial grammars [de Groote, 2001a], compositional continuation-based approaches to semantics that handle the problems related to quantification [Barker, 2002], donkey anaphora [de Groote, 2006], modal subordination [Asher and Pogodalla, 2010a]. This list is far from being exhaustive.

correspondance. The formalism is based on Girard's linear logic [Girard, 1987] and was created as a general framework capable of encoding other grammatical formalisms. The encoding has already been shown, for example, for tree adjoining grammars in [de Groote, 2002; Pogodalla, 2004], context-free formalisms (namely, context-free string grammars, linear context-free tree grammars, and linear context-free rewriting systems) in [de Groote and Pogodalla, 2004]. The power of abstract categorial grammar to encode other grammatical formalisms allows to make the comparison of their expressiveness.

> A logical framework does not consist of a single specific logic, but provides a means of defining various logics. It amounts to a general theory of logical systems that isolates the uniformities of a wide class of logics. Similarly, a grammatical framework should provide a means of defining various grammatical formalisms. [Philippe de Groote, ESSLLI 2009]

Abstract categorial grammars are similar to λ-grammars [Muskens, 2001] in that they express syntax-semantics correspondence in a parallel fashion.

Definition 1.24. [(Linear) implicative types] The set $\mathcal{T}(A)$ of **(linear) implicative types** built upon A is inductively defined as follows:

1. if $a \in A$, then $a \in \mathcal{T}(A)$
2. $\alpha, \beta \in \mathcal{T}(A)$, then $(\alpha \to \beta) \in \mathcal{T}(A)$.

Definition 1.25. [Higher-order (linear) signature] **Higher-order (linear) signature** Σ is a triple $\langle A, C, \tau \rangle$, where

1. A is a finite set of atomic types
2. C is a finite set of constants
3. $\tau : C \to \mathcal{T}(A)$ is a function that assigns to each constant in C a (linear) implicative type in $\mathcal{T}(A)$.

The type assignment τ imposes constraints on how the elements of vocabulary are combined to compose sentences.

Definition 1.26. [(Linear) λ-terms] Let X be an infinite countable set of λ-variables. The set $\Lambda(\Sigma)$ of **(linear) λ-terms** built upon a higher-order (linear) signature Σ, s.t. $\Sigma = \langle A, C, \tau \rangle$, is inductively defined as follows:

1. if $c \in C$, then $c \in \Lambda(\Sigma)$

2. if $x \in X$, then $x \in \Lambda(\Sigma)$
3. if $x \in X$, $t \in \Lambda(\Sigma)$, and x occurs free in t (exactly once), then $(\lambda x.t) \in \Lambda(\Sigma)$
4. if $t, u \in \Lambda(\Sigma)$, and the sets of free variables of t and u are disjoint, then $(tu) \in \Lambda(\Sigma)$.

DEFINITION 1.27. Given a higher-order (linear) signature Σ, s.t. $\Sigma = \langle A, C, \tau \rangle$, each (linear) λ-term in $\Lambda(\Sigma)$ may be assigned a (linear) implicative type in $\mathcal{T}(A)$. This type assignment obeys an inference system with sequents of the form $\Gamma \vdash_\Sigma t : \alpha$, where

1. Γ is a finite set of λ-variable typing declarations of the form "$x : \beta$" (with $x \in X$ and $\beta \in \mathcal{T}(A)$), such that any λ-variable is declared at most once
2. $t \in \Lambda(\Sigma)$
3. $\alpha \in \mathcal{T}(A)$.

The axioms and inference rules are the following:

$$\frac{}{\vdash_\Sigma c : \tau(c)} \text{ constant}$$

$$\frac{}{x : \alpha \vdash_\Sigma x : \alpha} \text{ variable}$$

$$\frac{\Gamma, x : \alpha \vdash_\Sigma t : \beta}{\Gamma \vdash_\Sigma (\lambda x.t) : (\alpha \to \beta)} \text{ abstraction (provided } x \text{ occurs free in } t \text{ exactly once)}$$

$$\frac{\Gamma \vdash_\Sigma t : (\alpha \to \beta) \qquad \Delta \vdash_\Sigma u : \alpha}{\Gamma, \Delta \vdash_\Sigma (tu) : \beta} \text{ application}$$

DEFINITION 1.28. [Vocabulary] A **vocabulary** is a higher-order (linear) signature.

DEFINITION 1.29. [Sentence] A **sentence** is a (linear) λ-term built upon a vocabulary.

DEFINITION 1.30. [Language] A **language** is the set of all sentences built upon a vocabulary.

Translations of a language into another are captured by the notion of lexicon:

DEFINITION 1.31. [Lexicon] Given two vocabularies Σ_1 and Σ_2, s.t. $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$, a **lexicon** $\mathcal{L}$ from Σ_1 to Σ_2 (i.e. $\mathcal{L} : \Sigma_1 \to \Sigma_2$) is defined as a pair $\langle F, G \rangle$ such that

1. $F : A_1 \to \mathcal{T}(A_2)$
2. $G : C_1 \to \Lambda(\Sigma_2)$
3. for any $c \in C_1$, the sequent $\vdash_{\Sigma_2} G(c) : \hat{F}(\tau_1(c))$, where $\hat{F}$ is the homomorphic extension of F, such that $\hat{F}(\alpha) \to \hat{F}(\beta) = \hat{F}(\alpha \to \beta)$, is derivable.

Definition 1.32. Let G be a function from C_1 to $\Lambda(\Sigma_2)$ in a lexicon $\langle F, G\rangle$. The homomorphic extension $\hat{G}(t)$ of G is a function from $\Lambda(\Sigma_1)$ to $\Lambda(\Sigma_2)$ defined as follows

1. $\hat{G} : C_1 \to \Lambda(\Sigma_2)$
2. if $x \in X$ and $x \in \Lambda(\Sigma_1)$, then $\hat{G}(x) = x$
3. $\hat{G}(\lambda x.t) = \lambda x.\hat{G}(t)$
4. $\hat{G}(tu) = \hat{G}(t)\hat{G}(u)$

The principle of compositionality can now be defined in a general abstract way:

Theorem 1.33. [Compositionality] *For any lexicon $\mathcal{L} : \Sigma_1 \to \Sigma_2$ with $\mathcal{L} = \langle F, G\rangle$ it is the case that if a sequent*

$$x_0 : \alpha_0, \ldots, x_n : \alpha_n \vdash_{\Sigma_1} t : \alpha$$

is derivable in Σ_1, then the sequent

$$x_0 : F(\alpha_0), \ldots, x_n : F(\alpha_n) \vdash_{\Sigma_2} \hat{G}(t) : \hat{F}(\alpha)$$

is derivable in Σ_2.

Proof. The proof is by induction on the structure of proofs.

- If t is a constant c, we have $\vdash_{\Sigma_1} c : \tau_1(c)$ and we have to prove $\vdash_{\Sigma_2} \hat{G}(c) : \hat{F}(\tau_1(c))$. By item 3 in Definition 1.31 we have $\vdash_{\Sigma_2} G(c) : \hat{F}(\tau_1(c))$. Then, by Definition 1.32 of homomorphic extension, $\hat{G}(c) = G(c)$, and hence we have $\vdash_{\Sigma_2} \hat{G}(c) : \hat{F}(\tau_1(c))$.
- If t is a variable x, we have $x : \alpha \vdash_{\Sigma_1} x : \alpha$ and we have to prove $x : \hat{F}(\alpha) \vdash_{\Sigma_2} \hat{G}(x) : \hat{F}(\alpha)$. By variable rule in Definition 1.27 we have $x : \hat{F}(\alpha) \vdash_{\Sigma_2} x : \hat{F}(\alpha)$. By Definition 1.32 of homomorphic extension we have $\hat{G}(x) = x$ and hence $x : \hat{F}(\alpha) \vdash_{\Sigma_2} \hat{G}(x) : \hat{F}(\alpha)$.

- If t is an abstraction $\lambda x.u$, we have

$$x_0 : \alpha_0, \ldots, x_n : \alpha_n \vdash_{\Sigma_1} \lambda x.u : \alpha \to \beta \tag{1.12}$$

and we have to prove

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(\lambda x.u) : \hat{F}(\alpha \to \beta)$$

The abstraction inference that has (1.12) as its conclusion has the following premise:

$$x_0 : \alpha_0, \ldots, x_n : \alpha_n, x : \alpha \vdash_{\Sigma_1} u : \beta \tag{1.13}$$

By induction hypothesis, if $x_0 : \alpha_0, \ldots, x_n : \alpha_n, x : \alpha \vdash_{\Sigma_1} u : \beta$, then $x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n), x : \hat{F}(\alpha) \vdash_{\Sigma_2} \hat{G}(u) : \hat{F}(\beta)$, and by modus ponens with (1.13), we have

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n), x : \hat{F}(\alpha) \vdash_{\Sigma_2} \hat{G}(u) : \hat{F}(\beta) \tag{1.14}$$

Applying abstraction rule to (1.14), we get

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \lambda x.\hat{G}(u) : \hat{F}(\alpha) \to \hat{F}(\beta)$$

By Item 3 in Definition 1.32 we have $\lambda x.\hat{G}(u) = \hat{G}(\lambda x.u)$ and by Item 3 in Definition 1.31 we have $\hat{F}(\alpha) \to \hat{F}(\beta) = \hat{F}(\alpha \to \beta)$. Hence,

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(\lambda x.u) : \hat{F}(\alpha \to \beta)$$

- If t is an application (uv), we have

$$x_0 : \alpha_0, \ldots, x_n : \alpha_n \vdash_{\Sigma_1} uv : \beta \tag{1.15}$$

and we have to prove

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(uv) : \hat{F}(\beta)$$

The application inference that has (1.15) as its conclusion has two premises:

$$x_0 : \alpha_0, \ldots, x_k : \alpha_k \vdash_{\Sigma_1} u : \alpha \to \beta \tag{1.16}$$

$$x_{k+1} : \alpha_{k+1}, \ldots, x_n : \alpha_n \vdash_{\Sigma_1} v : \alpha \tag{1.17}$$

By induction hypothesis, if $x_0 : \alpha_0, \ldots, x_k : \alpha_k \vdash_{\Sigma_1} u : \alpha \to \beta$, then $x_0 : \hat{F}(\alpha_0), \ldots, x_k : \hat{F}(\alpha_k) \vdash_{\Sigma_2} \hat{G}(u) : \hat{F}(\alpha \to \beta)$, and by modus ponens with (1.16), we have

$$x_0 : \hat{F}(\alpha_0), \ldots, x_k : \hat{F}(\alpha_k) \vdash_{\Sigma_2} \hat{G}(u) : \hat{F}(\alpha \to \beta)$$

By Item 3 in Definition 1.31, $\hat{F}(\alpha) \to \hat{F}(\beta) = \hat{F}(\alpha \to \beta)$, therefore

$$x_0 : \hat{F}(\alpha_0), \ldots, x_k : \hat{F}(\alpha_k) \vdash_{\Sigma_2} \hat{G}(u) : \hat{F}(\alpha) \to \hat{F}(\beta) \tag{1.18}$$

Moreover, by induction hypothesis, if $x_{k+1} : \alpha_{k+1}, \ldots, x_n : \alpha_n \vdash_{\Sigma_1} v : \alpha$, then $x_{k+1} : \hat{F}(\alpha_{k+1}), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(v) : \hat{F}(\alpha)$, and by modus ponens with (1.17), we have

$$x_{k+1} : \hat{F}(\alpha_{k+1}), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(v) : \hat{F}(\alpha) \tag{1.19}$$

Applying application rule to the sequents (1.18) and (1.19), we get

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(u)\hat{G}(v) : \hat{F}(\beta)$$

By Item 4 in Definition 1.32, we have $\hat{G}(u)\hat{G}(v) = \hat{G}(uv)$. Hence,

$$x_0 : \hat{F}(\alpha_0), \ldots, x_n : \hat{F}(\alpha_n) \vdash_{\Sigma_2} \hat{G}(uv) : \hat{F}(\beta)$$

□

Note that this formulation of the principle of compositionality is equivalent to the one defined in Section 1.2: Σ_1 defines the basic expressions of a given language, Σ_2 defines the basic expressions of some other language, the lexicon homomorphically interprets the sentences of the former language into the sentences of the latter.

Definition 1.34. [Abstract Categorial Grammar] An **abstract categorial grammar** (ACG) $\mathcal{G}$ is a quadruple $\langle \Sigma_0, \Sigma_1, \mathcal{L}, s \rangle$, where

1. Σ_0 and Σ_1 are two higher-order (linear) signatures $\langle A_0, C_0, \tau_0 \rangle$ and $\langle A_1, C_1, \tau_1 \rangle$ correspondingly
2. $\mathcal{L}$ is a lexicon, such that $\mathcal{L} : \Sigma_0 \to \Sigma_1$; Σ_0 is called the **abstract vocabulary**, Σ_1 is called the **object vocabulary**
3. $s \in \mathcal{T}(A_0)$ is a distinguished type of the grammar,

that generates two languages, the **abstract language** $\mathcal{A}(\mathcal{G})$, which is the set of closed (linear) λ-terms of type s built upon the abstract vocabulary Σ_0, and the **object language** $\mathcal{O}(\mathcal{G})$, which is the image of the abstract language by the term homomorphism induced by the lexicon $\mathcal{L}$, i.e.:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda(\Sigma_0) | \vdash_{\Sigma_0} t : s \text{ is derivable}\}$$

$$\mathcal{O}(\mathcal{G}) = \{t \in \Lambda(\Sigma_1) | \exists u \in \mathcal{A}(\mathcal{G}) : t = \mathcal{L}(u)\}$$

It is possible to express the relation between syntax and semantics of a language in abstract categorial grammar. For that two ACG sharing the same abstract language have to be defined. The object language of one of the grammars will stand for the source language, while the object language of another grammar will stand for the logical language representing semantics of the source language. Example 1.35 illustrates this.

EXAMPLE 1.35. Consider the following prominent sentence from [Montague, 1973]:

(10) *John seeks a unicorn.*

Below two ACGs, $\mathcal{G}_{syntax}$ and $\mathcal{G}_{semantics}$, with $\langle \Sigma_0, \Sigma_1, \mathcal{L}_1, s \rangle$ and $\langle \Sigma_0, \Sigma_2, \mathcal{L}_2, s \rangle$ respectively are defined. The abstract signature Σ_0 shared by both ACGs has three atomic types *n, np* and *s*; and four constants *J, U, A* and *S*, two of which have complex types:

$$\begin{aligned}\Sigma_0 = \langle &\{n, np, s\},\\ &\{J, U, A, S\},\\ &\{J \mapsto np,\\ &\ U \mapsto n,\\ &\ A \mapsto n \to ((np \to s) \to s),\\ &\ S \mapsto ((np \to s) \to s) \to (np \to s)\}\rangle\end{aligned}$$

The following λ-terms, denoted $t_{de\ dicto}$ and $t_{de\ re}$, can be built upon Σ_0:

$$\begin{aligned}t_{de\ dicto} &= S(AU)J\\ t_{de\ re} &= AU(\lambda x.S(\lambda k.kx)J)\end{aligned}$$

The object signature of $\mathcal{G}_{syntax}$ has only one type *string* and four constants of type *string*:

$$\begin{aligned}\Sigma_1 = \langle &\{string\},\\ &\{John, seeks, a, unicorn\},\\ &\{John \mapsto string,\\ &\ seeks \mapsto string,\\ &\ a \mapsto string,\\ &\ unicorn \mapsto string\}\rangle\end{aligned}$$

The lexicon $\mathcal{L}_1$ from Σ_0 to Σ_1 maps all types of Σ_0 into the type *string* of Σ_1, the constants J and U are mapped into the constants John and unicorn correspondingly, the constants A and S are mapped into complex terms:

$$\begin{aligned}\mathcal{L}_1 = \langle\{&n \mapsto \mathit{string}, \mathit{np} \mapsto \mathit{string}, s \mapsto \mathit{string}\},\\ &\{J \mapsto \mathit{John},\\ &\ U \mapsto \mathit{unicorn},\\ &\ A \mapsto \lambda xp.p(a + x),\\ &\ S \mapsto \lambda px.p(\lambda y.x + \mathit{seeks} + y)\}\rangle\end{aligned}$$

The syntactic lexicon $\mathcal{L}_1$ applied to the terms $t_{de\ dicto}$ and $t_{de\ re}$ results in terms that are identical after β-reduction. These terms represent Sentence (10):

$$\begin{aligned}\mathcal{L}_1(t_{de\ dicto}) &\rightarrow^*_\beta \mathit{John} + \mathit{seeks} + a + \mathit{unicorn}\\ \mathcal{L}_1(t_{de\ re}) &\rightarrow^*_\beta \mathit{John} + \mathit{seeks} + a + \mathit{unicorn}\end{aligned}$$

The object signature of $\mathcal{G}_{semantics}$ has two atomic types ι and o, one constant of type ι and five constants of complex types:

$$\begin{aligned}\Sigma_2 = \langle\{&\iota, o\},\\ &\{\mathbf{j}, \mathbf{unicorn}, \mathbf{find}, \mathbf{try}, \wedge, \exists\},\\ &\{\mathbf{j} \mapsto \iota,\\ &\ \mathbf{unicorn} \mapsto \iota \rightarrow o,\\ &\ \mathbf{find} \mapsto \iota \rightarrow (\iota \rightarrow o),\\ &\ \mathbf{try} \mapsto \iota \rightarrow ((\iota \rightarrow o) \rightarrow o)\},\\ &\ \wedge \mapsto o \rightarrow (o \rightarrow o),\\ &\ \exists \mapsto (\iota \rightarrow o) \rightarrow o\}\rangle\end{aligned}$$

The lexicon $\mathcal{L}_2$ maps types and terms from Σ_0 to Σ_2:

$$\begin{aligned}\mathcal{L}_2 = \langle\{&n \mapsto (\iota \rightarrow o), \mathit{np} \mapsto \iota, s \mapsto o\},\\ &\{J \mapsto \mathrm{j},\\ &\ U \mapsto \lambda x.\mathbf{unicorn}x,\\ &\ A \mapsto \lambda pq.\exists x.px \wedge qx,\\ &\ S \mapsto \lambda px.\mathbf{try}x(\lambda y.p(\lambda z.\mathbf{find}yz))\}\rangle\end{aligned}$$

After applying the lexicon $\mathcal{L}_2$ to the terms $t_{de\ dicto}$ and $t_{de\ re}$, and β-reducing, the corresponding de dicto and de re readings of Sentence (10) are obtained:

$$\begin{aligned}\mathcal{L}_2(t_{de\ dicto}) &\rightarrow^*_\beta \mathbf{try}\ \mathbf{j}(\lambda x.\exists y.\mathbf{unicorn}y \wedge \mathbf{find}xy)\\ \mathcal{L}_2(t_{de\ re}) &\rightarrow^*_\beta \exists y.\mathbf{unicorn}y \wedge \mathbf{try}\ \mathbf{j}(\lambda x.\mathbf{find}xy)\end{aligned}$$

□

Figure 1.7 schematically represents how syntax and semantics of a (natural) language are related through two abstract categorial grammars sharing an abstract vocabulary. Note, that the translation from a language to a logical language is indirect via an abstract syntactic structure. Namely, it is a relational composition of the inverse of the syntactic lexicon and the semantic lexicon, as schematically depicted in Figure 1.8. Example 1.36, based on Example 1.35, illustrates this in more details.

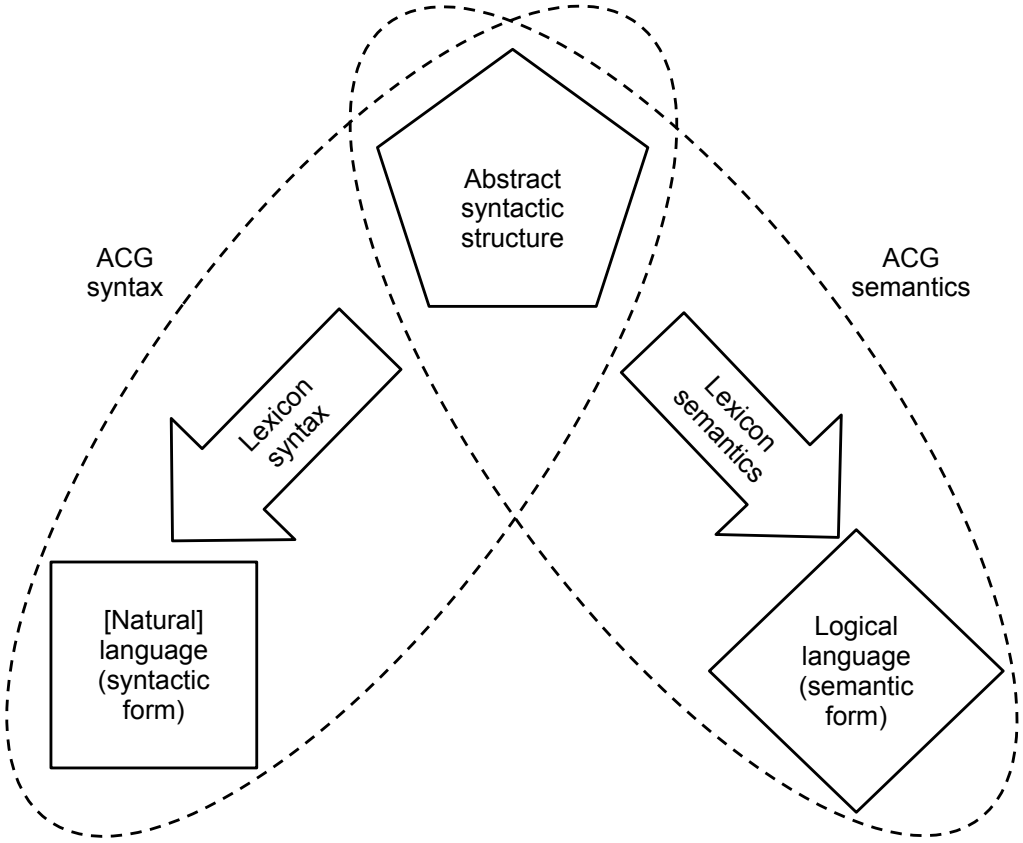


Figure 1.7: Relation of syntax and semantics of a language through two ACG sharing abstract vocabulary.

Example 1.36. Let $\mathcal{G}_{syntax}$ and $\mathcal{G}_{semantics}$ be two abstract categorial grammars defined in Example 1.35. Then, the inverse image $\mathcal{L}_1^{-1}$ of $\mathcal{L}_1$ is as follows:

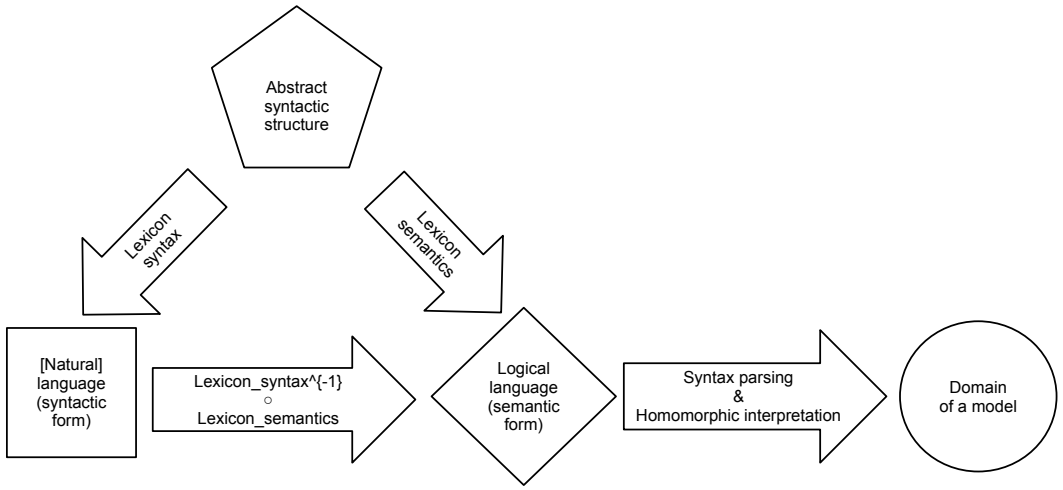


Figure 1.8: Syntax, semantics and model-theoretic interpretation of a language.

$$\begin{aligned}\mathcal{L}_1^{-1}(John) &= J\\ \mathcal{L}_1^{-1}(unicorn) &= U\\ \mathcal{L}_1^{-1}(a) &= \lambda x.Ax,\\ \mathcal{L}_1^{-1}(seeks) &= \lambda xy.Sxy\end{aligned}$$

Applying $\mathcal{L}_1^{-1}$ to the sentence "$John + seeks + a + unicorn$" built upon the vocabulary Σ_1 can yield $t_{de\ dicto}$ and $t_{de\ re}$:

$$\mathcal{L}_1^{-1}(John + seeks + a + unicorn) = S(AU)J \tag{1.20}$$
$$\mathcal{L}_1^{-1}(John + seeks + a + unicorn) = AU(\lambda x.S(\lambda k.kx)J) \tag{1.21}$$

Then, applying $\mathcal{L}_2$ to $t_{de\ dicto}$ and $t_{de\ re}$ yields, as already shown in Example 1.35, de dicto and de re readings of the initial sentence:

$$\mathcal{L}_2(t_{de\ dicto}) \to_\beta^* \mathbf{try}\ \mathbf{j}(\lambda x.\exists y.\mathbf{unicorn}y \wedge \mathbf{find}xy)$$
$$\mathcal{L}_2(t_{de\ re}) \to_\beta^* \exists y.\mathbf{unicorn}y \wedge \mathbf{try}\ \mathbf{j}(\lambda x.\mathbf{find}xy)$$

It is important to note that $\mathcal{L}_1^{-1}$ is not a function. Applying $\mathcal{L}_1^{-1}$ to the same term may result in more than one possible interpretation, as happened in (1.20) and (1.21).

Definition 1.37. [Logical Meaning] $[\![\cdot]\!]$ denotes $\mathcal{L}_1^{-1} \circ \mathcal{L}_2$ and is called **logical meaning**.

Therefore, two possible logical meanings of the sentence (10) are computed:

$$[\![\mathit{John} + \mathit{seeks} + a + \mathit{unicorn}]\!] = \mathbf{try}\ \mathbf{j}(\lambda x.\exists y.\mathbf{unicorn}y \wedge \mathbf{find}xy)$$
$$[\![\mathit{John} + \mathit{seeks} + a + \mathit{unicorn}]\!] = \exists y.\mathbf{unicorn}y \wedge \mathbf{try}\ \mathbf{j}(\lambda x.\mathbf{find}xy)$$

□

Clearly, as discussed in Section 1.4, the logical language can be subsequently model-theoretically interpreted: compare Figures 1.8 and 1.5.

This chapter introduced the basics necessary for understanding interpretation of natural language, such as truth-conditionality, compositionality and model-theory; and it presented, based on Montague's theory, the fundamentals of formal semantics of natural language as a procedure of interpreting a natural language into a logical language. The rest of the thesis is concerned with defining a formal mathematically precise compositional semantics in the spirit of Montague semantics capable of handling (at least some) natural language dynamic phenomena. The next chapter presents these phenomena and the challenges they bring to compositional semantics.

Chapter 2

Natural Language Dynamics

Despite Montague's certainty that natural and formal languages can be similarly treated, and the evidence that this is the case (at least) for fragments of natural language he presented in [1970a; 1970b; 1973] where he considered important issues as scope ambiguity and intensionality, his semantics still needed to be extended in order to cope with other complex natural language phenomena.

This chapter is concerned with phenomena of **discourse dynamics**, including cross-sentential pronominal and donkey anaphora (Subsections 2.1.2 and 2.1.3 respectively), presuppositions (Subsection 2.1.4) and conversational implicatures (Subsection 2.1.5). The common characteristic of these phenomena is that they are, in one way or another, dependent upon a **context** (Subsection 2.1.1), i.e. some preliminary knowledge or a common ground. Section 2.2 presents the currently most used dynamic theories, namely Discourse Representation Theory, Dynamic Predicate Logic and Dynamic Montague Grammar, and discusses how they deal with (some) of the dynamic phenomena.

2.1 Motivation for a Dynamic Theory

The necessity to interpret not only single sentences, but also discourses, lead to the evolution of Montague's analogy between natural and formal languages to the analogy between natural languages and (formal) computer programs. Each sentence (utterance) of a discourse has been paralleled with a statement of a program; and the interpretation of the whole discourse has been paralleled with the evaluation of the whole computer program.

Consider, for example, the tiny program (2.1a), where x is a variable. To evaluate this program, x must be firstly provided some value. For example, the program in (2.1b), where x is assigned the value 2 by the statement $x := 2;$, prints 2. However, the program in (2.1c), where x is assigned the value 3 by a different statement $x := 3;$, prints 3. Thus, although the second statements of programs

in (2.1b) and (2.1c) are identical, i.e. $\mathsf{Print}(x)$, the difference in first statements in (2.1b) and in (2.1c) results in different evaluations of $\mathsf{Print}(x)$. In other words, the second statement in these programs is, in one way or another dependent, on the respective first statements.

$$\mathsf{Print}(x); \tag{2.1a}$$
$$x := 2; \mathsf{Print}(x); \tag{2.1b}$$
$$x := 3; \mathsf{Print}(x); \tag{2.1c}$$

Consider now the short sentence (11a). If it is uttered alone, it is not clear to whom *he* refers to. However, if it is preceded by another sentence, for example as in (11b), it is evident that *he* refers to the man walking in the park. On the other hand, if the preceding sentence is different, as in (11c), *he* refers to a (possibly) different individual: John. Therefore, although the second sentences in discourses (11b) and (11c) are syntactically identical, they can lead to different meanings depending on the respective first sentences.

(11) a. *He whistles.*

b. *A man walks in the park. He whistles.*

c. *John supports his soccer team in the stadium. He whistles.*

The analogy between natural and programming languages helps to explain the order-sensitivity of natural language expressions. For example, it explains why, while it is fine to say (12a), switching the order of the sentences leads to an infelicitous discourse (12b):[1]

(12) a. *A man walks in the park. He whistles.*

b. *He whistles. A man walks in the park.*

Although dynamics in natural language was noticed even before the proliferation of programming languages [Lewis, 1979], it became more recognized with the development of semantics of programming languages. Since dynamic phenomena in programming languages were successfully treated by the use of dynamic logics (e.g. [Harel *et al.*, 1984]), the analogy between natural and programming languages gave a stronger motivation to dynamically handle discourse phenomena in semantics of natural language. This resulted in emergency of various dynamic theories and approaches, including representational theories with a particular emphasis on context [Kamp, 1981; Heim, 1982], theories modifying semantics of programming languages to fit natural language meaning [Groenendijk and Stokhof, 1990], theories combining the advantages of the previous two [Muskens, 1996], theories based on game-theoretic semantics [Hintikka and Sandu, 1997], frameworks

[1] (12a) is borrowed from [Groenendijk and Stokhof, 1991] or [Gamut, 1991].

elaborating the analogy between computational and linguistic side effects [Shan, 2005].

Approaches that incorporate the dependency of the meaning of the sentence on previous sentences or previously established context are of particular interest to this dissertation. A brief analysis of such theories is given in Section 2.2. The immediate subsection explains what a context is and gives a historical provision of incorporating the notion of context into formal semantics. Subsections 2.1.2– 2.1.5 discuss some particular (and challenging) context-dependent dynamic phenomena of natural language.

2.1.1 Context

Informally, **context** is (an abstract representation of) the current knowledge of an agent, including everything that she/he has learned (and not forgotten) since birth. Every time a person encounters a new piece of information, she/he analyses this new information with respect to her/his context and, depending on the analysis, modifies the context. There are numerous sources of new information for a human being. Being concerned with formal semantics, this dissertation is focused on knowledge provided by means of natural language expressions such as sentences and discourses.

As Jaroslav Peregrin [2003] notes, the initial logico-philosophical approaches to meaning (by Wittgenstein, Russel and Carnap) disregarded contexts of use of natural language, and strived for a way to express meaning as an autonomous abstract system. Moreover, context-dependence was sometimes considered as an imperfection of natural language. To study the ways in which context contributes to meaning has been the duty of another field of research, **pragmatics**. However, with the Montague's precise formalization of meaning and subsequent rapid development of formal semantics, it became clear that the fields of formal semantics and pragmatics should work in tandem, and that a sentence has to be formally interpreted with respect to a context. In fact, Montague himself foresaw this:

> It seemed to me desirable that pragmatics should at least initially follow the lead of semantics, which is primarily concerned with the notion of truth (in a model, or under an interpretation), and hence concern itself also with truth - but with respect not only to an interpretation but also to a context of use. [Montague, 1968]

Context as a whole seems to be a very complex structure having various components including not only already acquired knowledge, but also place, time, environment, ... This, for example, has already been observed by David Lewis in his pioneering paper on formal semantics:

> [...] we must have several *contextual coordinates* corresponding to familiar sorts of dependence on features of context. (The world

> coordinate itself might be regarded as a feature of context, since different possible utterances of a sentence are located in different possible worlds.) We must have a *time coordinate*, in view of tensed sentences and such sentences as 'Today is Tuesday'; a *place coordinate*, in view of such sentences as 'Here there are tigers'; a *speaker coordinate* in view of such sentences as 'I am Porky'; an *audience coordinate* in view of such sentences as 'You are Porky'; an *indicated-objects coordinate* in view of such sentences as 'That pig is Porky' or 'Those men are Communists'; and a *previous discourse coordinate* in view of such sentences as 'The aforementioned pig is Porky'. [Lewis, 1970, p.24]

According to [Peregrin, 2003], the first systematic accounts of context dependence are given by David Kaplan [1970] and Robert Stalnaker [1970; 1978]. At that time possible world semantics was a prevailing theory of language interpretation and, therefore, as Lewis' semantics theory, Kaplan's and Stalnaker's accounts of context-dependency were given in terms of possible worlds.[2] Focusing on demonstratives, Kaplan [1970] criticised viewing the intention of an expression as a function not only from possible worlds, but also from a context to the extension.[3] He argued that contexts and possible worlds play very distinct roles and should not be fused. He considered intension, as a function from possible worlds to extensions, to be the content of an expression and he urged that it is the content that is dependent on the context:

> The content of an expression is always taken *with respect to* a given context of use. Thus when I say
>
> (4) I was insulted yesterday.
>
> a specific content - *what I said* - is expected. Your utterance of the same sentence, or mine on another day, would not express the same content. What is important to note is that it is not just the truth value that may change; what is said is itself different. Speaking today, my utterance of (4) will have a content roughly equivalent to that which
>
> (5) David Kaplan is insulted on April 20, 1973.

[2] Possible worlds were introduced in the middle of 20th century [Hintikka, 1962; Kripke, 1963] to define semantics for modal logics, which study deductive behaviour of the expressions "it is necessary that" (knowledge) and "it is possible that" (belief). Due to the existence of analogous modalities in natural language, natural language semantics adopted possible worlds approach. In possible world semantics, a valuation function assigns a truth value to each proposition with respect to each possible world: a propositional variable can be assigned different values in different worlds.

[3] The notions of "intension" and "extension" are often used to express the same aspects as Fregean "sense" and "reference" respectively (recall the discussion of relativity of truth with respect to a model in Section 1.3). Intensions and extensions are usually characterized in terms of possible world semantics. For example, intensional expressions are interpreted differently in different worlds. See [Fitting,] for an overview.

> would have spoken by you or anyone at any time. Since (5) contains no demonstratives, its content is the same with respect to all contexts. [Kaplan, 1970, p.83-84]

Therefore, according to Kaplan, a sentence has to be necessarily evaluated with respect to a context in order to unfold the indexical; then the resulting proposition can be interpreted w.r.t. some possible worlds to give an extension.

Stalnaker [1970] shared the view that "the linguistic context determines the proposition expressed by a given sentence in that context" and that "contextual determinants of propositions and propositional determinants of truth" should be distinguished.[4] Moreover, he went a step further and argued that sentences (or assertions) are not only confronted with the context, but they also change it. This view clearly reflected a new dynamic tendency.

> Let me begin with some truisms about assertions. First, assertions have content; an act of assertion is, among other things, the expression of a proposition - something that represents the world as being a certain way. Second, assertions are made in context - a situation that includes a speaker with certain beliefs and intentions, and some people with their own beliefs and intentions to whom the assertion is addressed. Third, sometimes the content of the assertion is dependent on the context in which it is made, for example, on who is speaking or when the act of assertion takes place. Fourth, acts of assertion affect, and are intended to affect, the context, in particular the attitudes of the participants in the situation: how the assertion affects the context will depend on its content. [Stalnaker, 1978, p.315]

The (potential) ability of sentences to take a context and modify it received later a self-explanatory name, **context change potential** [Heim, 1982], and a new perspective where the main function of a sentence was to update the context and the actual (asserted) content of the sentence was seen as a mean for this update. This led to identifying meanings of sentences with their context change potentials. Although this simplification led to significant progress in the study of discourse semantics [Heim, 1982; Heim, 1983; Kamp, 1981; Kamp and Reyle, 1993; Groenendijk and Stokhof, 1990; Groenendijk and Stokhof, 1991],[5] the neglect of the actual content of sentences (and discourses) makes it impossible to model-theoretically interpret them (i.e. evaluating whether what is said is true

[4] In [1978] Stalnaker refers to Wittgenstein's Tractatus Logico-Philosophicus [Wittgenstein, 1921] and uses his terminology in explaining the assumption that the sense of a sentence is not dependent on the truth value of another proposition. He clarifies that the phenomenon of context dependence is an evidence for this claim.

[5] Heim's file change semantics and Kamp's discourse representation theory are two independent frameworks that, however, are based on similar fundamental principles. Subsection 2.2.1 thus focuses only on discourse representation theory, which is nowadays more elaborated.

or false in some possible situation) independently of a context. Stalnaker have recently commented on this problem:

> Meaning determines the content of an assertion as a function of context, and the assertion rule takes the prior context set to a posterior context set, which is the intersection of the prior set with that content. Some of the dynamic semantic theories subsequently developed by linguists have blurred the distinction between content and force by combining the two steps (meaning plus prior context to content, and prior context plus content to posterior context) into one. Irene Heim, for example, proposed to represent the meaning of a sentence as its *context-change potential* [Heim(1982)], which is a function taking the prior context directly to the posterior context. I think this streamlined representation captures much of what is important about the dynamic process of speech, but what it leaves out is the possibility of evaluating the truth or falsity of what is said relative to possible situations that are not compatible with the prior context. Sometimes when a statement rests on false presuppositions, the question of the actual truth of the statement does not arise, but other times a speaker may succeed in making a claim that is actually true or false, even when taking for granted, in making the claim, something that is in fact false. In such cases, our semantic theory should tell us that is said, and not just how what is said changes the context. Sentences that say different things in some contexts may nevertheless change contexts in the same way. [Stalnaker, 1999, p.11].

This issue is further discussed at the end of the chapter, after the relevant dynamic theories are presented.

2.1.2 Cross-Sentential Pronominal Anaphora

An anaphoric expression can be defined in the following general way:

Definition 2.1. [Anaphoric expression] An **anaphoric expression** (**anaphor**) is an expression the meaning of which depends on the meaning of another expression (called **antecedent**) contained in a context.

Cross-sentential pronominal anaphora is one of the most common cases of the context-dependence. Recall discourses (11b) and (11c), repeated below:

(13) a. *A man walks in the park. He whistles.*

b. *John supports his soccer team in the stadium. He whistles.*

Assume each of (13) is independently uttered in the same context c. Following a natural assumption that discourse is processed incrementally, for each of the discourses in (13), c has to be first updated with the respective first sentences, and than the new context has to be updated with the second sentences. Assume that updating c with *A man walks in the park* resulted in a new context c' and updating c with *John supports his soccer team in the stadium* resulted in a new context c''. c' is different from c'', because they resulted from updating c with different sentences. Now, when *He whistles* is evaluated with respect to c' and c'', the pronoun *he* is associated with person with different descriptive content: with an individual who is a man walking in park, in the case of c'; and with an individual whose name is "John" and who supports his soccer team in the stadium, in the case of c''. This illustrates that the meaning of the anaphoric pronoun *he* is relative to the context in which it occurs.

The phenomenon of cross-sentential pronominal anaphora is one of the challenges for formal semantics of natural language, because, among other issues, the formalization of the anaphoric link between a pronoun in one of the sentences and a noun phrase from a preceding sentence is not trivial. Consider again the short discourse (13a). Assuming that the composition of sentences of a discourse is interpreted as a logical conjunction, the desired interpretation of the whole discourse (13a) in first order logic would be as represented in Formula (2.2), where the existentially quantified variable is introduced by the indefinite article:

$$\exists x.\mathbf{man}\ x \wedge \mathbf{walks_in_the_park}\ x \wedge \mathbf{whistles}\ x \tag{2.2}$$

The challenge is to build (2.2) compositionally, from the interpretations of the two sentences composing (13a). Assuming that the indefinite article introduces an existentially quantified variable (which is a standard and natural assumption in natural language semantics), it is not difficult to translate the first sentence, *A man walks in the park*, into the first order logical formula (2.3):

$$\exists x.\mathbf{man}\ x \wedge \mathbf{walks_in_the_park}\ x \tag{2.3}$$

The matter is, however, harder for the second sentence. Being a pronoun, *he* should be interpreted into something that can somehow refer to its antecedent (e.g., in this case to a man walking in the park), or, even better, retrieve the antecedent from the context. For the time being, *he* is interpreted tentatively as a variable with a superscript "?", indicating that the variable awaits to be linked with its antecedent. Then, the logical formula for the second sentence can be represented as (2.4):

$$\mathbf{whistles}\ y^{?} \tag{2.4}$$

Having logical formulas for both sentences of (13a), their composition with the logical conjunction (as (temporarily) agreed above), results in the following

formula:

$$(\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x) \wedge \textbf{whistles}\ y^{?} \tag{2.5}$$

It remains to find the antecedent of $y^{?}$ and link $y^{?}$ with it. However, to do so it is necessary to answer at least the following questions:

How should the anaphora be resolved? In other words, which method should be used for finding the antecedent of the anaphoric expression?

How should the "link" between the antecedent and an anaphor be defined? One possibility is to use a unique choice operator, ι, to build an **epsilon term** and assign it to the anaphoric variable. In the example, $y^{?}$ would be assigned the epsilon term $\iota_y(\textbf{man}\ y \wedge \textbf{walks_in_the_park}\ y)$ denoting the (unique) man who walks in the park.[6] However, the logical formula obtained with epsilon-approach appears to be redundant (2.6):

$$\begin{aligned}&(\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x)\wedge\\ &\textbf{whistles}(\iota_y(\textbf{man}\ y \wedge \textbf{walks_in_the_park}\ y))\end{aligned} \tag{2.6}$$

Another[7], more concise, possibility is to assign the interpretation of the antecedent to the variable standing for the anaphor. Then, since existentially quantified x already represents a man walking in the park, it would be directly assigned to $y^{?}$ in (2.5), resulting in the formula (2.7):

$$(\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x) \wedge \textbf{whistles}\ x \tag{2.7}$$

However, this approach has a scope problem. Formula (2.7) is not equivalent to the desired Formula (2.2) of interpretation of the discourse. In fact, (2.7) is equivalent to (2.5) and the substitution of the anaphoric variable $y^{?}$ with the variable x standing for the antecedent did not have any effect from logical point of view. The reason is that $y^{?}$ in (2.5) is not within the scope of the quantifier that binds x. Hence, when $y^{?}$ is substituted with x, this (substituted) x is also not bound. Therefore, there is no advance from (2.5).

The scope problem leads to the next question: **If the antecedent is a quantified variable, how can an anaphoric variable and its predicates fall inside the quantifier scope?** The scope problem would not happen if **whistles** $y^{?}$ were inside the scope of the existential quantifier. In other words, if, instead of Formula (2.5) we could get (2.8), the substitution of $y^{?}$ for variable x standing for the antecedent, would lead to the desired interpretation (2.2):

$$(\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x \wedge \textbf{whistles}\ y^{?}) \tag{2.8}$$

[6] For more details on how choice operators are used to analize anaphoric pronouns, definite and indefinite descriptions, see [Egli and von Heusinger, 1995; von Heusinger, 1997].

[7] Analyses of other, less steady, approaches can be found in [Heim, 1982, ch.1].

Answers to the preceding questions would be prerequisites to answering a more general question: **What is the definition of the composition of sentences into a discourse?** Formula (2.5) is the result of the composition of two sentences, which is simply defined as a logical conjunction. The example shows that the composition should be defined in a more elaborated way.

2.1.3 Donkey Sentences

Donkey sentences are sentences that have a particular type of anaphora which had already been discussed by Stoic philosopher Chrysippus, according to [Heim, 1982, p.44]. The name "donkey sentences" is, however, due to the examples of Geach [1962], who revived the interest to this type of sentences in modern philosophy.

The most popular examples of donkey sentences are as follows:

(14) a. *If a farmer owns a donkey, he beats it.*

b. *Every farmer who owns a donkey beats it.*

Sentence (14a) contains an indefinite noun phrase (*a donkey*) which is inside the antecedent ("if"-clause) of a conditional sentence and a pronoun (*it*) which is inside the consequent ("then"-clause) of the sentence but is related anaphorically to the indefinite noun phrase.

Sentence (14b) contains an indefinite noun phrase (*a donkey*) which is inside a relative clause and a pronoun (*it*) which is outside the relative clause but is related anaphorically to the indefinite noun phrase. Moreover, the sentence has a quantified noun phrase (*every farmer*) and the pronoun (*it*) is in a relation with individuals denoted by this noun phrase.

The most generally accepted interpretation of Sentences (14) in first order logic is (2.9), which can be spelled as "for every farmer and every donkey, if he owns it, then he beats it":

$$\forall xy.(\mathbf{farmer}\ x \wedge \mathbf{donkey}\ y \wedge \mathbf{owns}\ x\ y \rightarrow \mathbf{beats}\ x\ y) \tag{2.9}$$

However, to construct Formula (2.9) compositionally from (14a) or (14b) is a non-trivial task. The reasons are explained below for Sentence (14b), Sentence (14a) causes similar challenges.

First of all, there is a pronominal anaphor in (14b) and it is analogous to the cross-sentential anaphora, because the antecedent (*a donkey*) of the pronoun (*it*) occurs in the relative clause. Therefore, the scope problem of cross-sentential anaphora takes place in donkey sentences as well. Particularly, the sub-formulas interpreting the expressions outside the relative clause should be within the scope of the existential quantifier introduced by an indefinite article which occurs inside the relative clause. Thus, all the questions discussed in Subsection 2.1.2 are applicable to the donkey sentences.

Turning to the example, an attempt to give a naive straight-forward interpretation of Sentence (14b) in first order logic, can result in Formula (2.10):

$$\forall x.(\textbf{farmer}\ x \wedge \exists y.(\textbf{donkey}\ y \wedge \textbf{owns}\ x\ y) \rightarrow \textbf{beats}\ x\ v^{?}) \tag{2.10}$$

Simply assigning y to $v^{?}$ will not logically change the formula, because $v^{?}$ is outside the scope of the existential quantifier binding y. The following questions arise: **How can the formulas occuring outside a clause where a quantifier was triggered fall into the scope of this quantifier?**[8] **Under which conditions does it happen? What are the factors influencing the quantifier scope?**

Moreover, a simple extension of the quantifier scope leads to an incorrect interpretation. Suppose the scope of the existential quantifier in Formula (2.10) was extended and $v^{?}$ was assigned the bound variable y, as shown in Formula (2.11):

$$\forall x \exists y.(\textbf{farmer}\ x \wedge \textbf{donkey}\ y \wedge \textbf{owns}\ x\ y \rightarrow \textbf{beats}\ x\ y) \tag{2.11}$$

Formula (2.11) is not logically equivalent to the desired Formula (2.9). For example, imagine a model in which there is only one farmer having two donkeys, such that the farmer beats one of his donkeys and never beats the other one. Formula (2.11) is true and Formula (2.9) is false in such a model. To get the correct interpretation of the sentence, the existential quantifier should be changed to a universal quantifier. This leads to the question: **What exactly causes the change of the quantifier?**

To provide a better intuition about the problems formulated above, the compositional computation of the meaning of the donkey sentence (14b) using classical static lexical interpretations (shown in Table 2.1[9]) in the spirit of Montague is further demonstrated in detail.

According to the parse tree, shown in Figure 2.1, the static meaning of Sentence (14b) can be computed by β-reducing the term (2.12):

$$[\![beats]\!][\![it]\!]([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!])) \tag{2.12}$$

[8] This is often expressed as "the quantifier extends its scope". Saying so is deliberately avoided here, since the quantifier itself does not have any scope-extending power. The extension of the scope is rather influenced by different factors.

[9] In order to make the computation of meanings more concise, the constants are abbreviated with their first letters. For example, the constant **donkey** is abbreviated as **d**.

Lexical item	Syntactic category	Static interpretation
farmer	n	$\mathbf{f}$
donkey	n	$\mathbf{d}$
owns	$\mathrm{np} \to \mathrm{np} \to \mathrm{s}$	$\lambda \mathrm{YX.X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{o}xy))$
beats	$\mathrm{np} \to \mathrm{np} \to \mathrm{s}$	$\lambda \mathrm{YX.X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{b}xy))$
a	$\mathrm{n} \to \mathrm{np}$	$\lambda \mathrm{PQ}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)$
every	$\mathrm{n} \to \mathrm{np}$	$\lambda \mathrm{PQ}.\forall(\lambda x.\mathrm{P}x \to \mathrm{Q}x)$
who	$(\mathrm{np} \to \mathrm{s}) \to \mathrm{n} \to \mathrm{n}$	$\lambda \mathrm{RQ}x.\mathrm{Q}x \wedge \mathrm{R}(\lambda \mathrm{P.P}x)$
it	np	$\lambda \mathrm{P.P}v^?$, where $v^?$ is a fresh variable

Table 2.1: Static interpretations of lexical items of the sentence *Every farmer who owns a donkey beats it.*

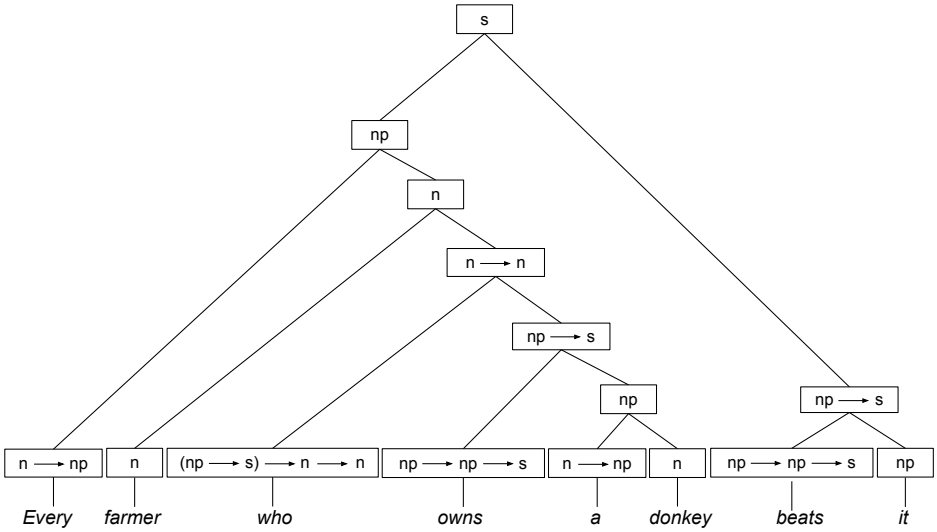


Figure 2.1: Syntactic parse tree of sentence *Every farmer who owns a donkey beats it.*

The meaning of the noun phrase *a donkey* is computed by β-reducing the term $[\![a]\!][\![donkey]\!]$:

$$\begin{aligned}[\![a]\!][\![donkey]\!] &= (\lambda \mathrm{PQ}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x))[\![donkey]\!] \\ &\to_\beta \lambda \mathrm{Q}.\exists(\lambda x.[\![donkey]\!]x \wedge \mathrm{Q}x) \\ &= \lambda \mathrm{Q}.\exists(\lambda x.\mathbf{d}x \wedge \mathrm{Q}x)\end{aligned}$$

The term $[\![owns]\!]([\![a]\!][\![donkey]\!])$ is β-reduced in the following way:

$$\begin{aligned}[\![owns]\!]([\![a]\!][\![donkey]\!]) &= (\lambda \mathrm{YX.X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{o}xy)))([\![a]\!][\![donkey]\!]) \\ &\to_\beta \lambda \mathrm{X.X}(\lambda x.[\![a]\!][\![donkey]\!](\lambda y.\mathbf{o}xy))\end{aligned}$$

$$\begin{aligned}
&= \lambda \mathrm{X}.\mathrm{X}(\lambda x.(\lambda \mathrm{Q}.\exists(\lambda z.\mathbf{d}z \wedge \mathrm{Q}z))(\lambda y.\mathbf{o}xy)) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge (\lambda y.\mathbf{o}xy)z)) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz))
\end{aligned}$$

The meaning of the relative clause *who owns a donkey* is computed by β-reducing the term $[\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))$:

$$\begin{aligned}
[\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])) &= (\lambda \mathrm{RQ}y.\mathrm{Q}y \wedge \mathrm{R}(\lambda \mathrm{P}.\mathrm{P}y))([\![owns]\!]([\![a]\!][\![donkey]\!])) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge [\![owns]\!]([\![a]\!][\![donkey]\!])(\lambda \mathrm{P}.\mathrm{P}y) \\
&= \lambda \mathrm{Q}y.\mathrm{Q}y \wedge (\lambda \mathrm{X}.\mathrm{X}(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)))(\lambda \mathrm{P}.\mathrm{P}y) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge (\lambda \mathrm{P}.\mathrm{P}y)(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge (\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)y) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz)
\end{aligned}$$

The meaning of *farmer who owns a donkey* is computed by applying the resulting λ-term in the computation above to the interpretation of *farmer*:

$$\begin{aligned}
([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!] &= (\lambda \mathrm{Q}y.\mathrm{Q}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz))[\![farmer]\!] \\
&\rightarrow_\beta \lambda y.[\![farmer]\!]y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz) \\
&= \lambda y.\mathbf{f}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz)
\end{aligned}$$

The meaning of the noun phrase *every farmer who owns a donkey* is computed by applying the interpretation of *every* to the interpretation of *farmer who owns a donkey*:

$$\begin{aligned}
&[\![every]\!]([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]) \\
&= (\lambda \mathrm{PQ}.\forall(\lambda x.\mathrm{P}x \rightarrow \mathrm{Q}x))([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]) \\
&\rightarrow_\beta \lambda \mathrm{Q}.\forall(\lambda x.([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]x \rightarrow \mathrm{Q}x) \\
&= \lambda \mathrm{Q}.\forall(\lambda x.(\lambda y.\mathbf{f}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz))x \rightarrow \mathrm{Q}x) \\
&\rightarrow_\beta \lambda \mathrm{Q}.\forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \rightarrow \mathrm{Q}x) \qquad (2.13)
\end{aligned}$$

The meaning of the verb phrase *beats it* is computed as follows:

$$\begin{aligned}
[\![beats]\!][\![it]\!] &= (\lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{b}xy)))[\![it]\!] \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.[\![it]\!](\lambda y.\mathbf{b}xy)) \\
&= \lambda \mathrm{X}.\mathrm{X}(\lambda x.(\lambda \mathrm{P}.\mathrm{P}v^?)(\lambda y.\mathbf{b}xy)) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.(\lambda y.\mathbf{b}xy)v^?) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.\mathbf{b}xv^?) \qquad (2.14)
\end{aligned}$$

Finally, the static meaning of the sentence is computed by applying the term (2.14) to the term (2.13):

$$\begin{aligned}
&[\![beats]\!][\![it]\!]([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!])) \\
&= (\lambda \mathrm{X}.\mathrm{X}(\lambda x.\mathbf{b}xv^?))([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!])) \\
&\rightarrow_\beta ([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]))(\lambda x.\mathbf{b}xv^?) \\
&= (\lambda \mathrm{Q}.\forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \rightarrow \mathrm{Q}x))(\lambda x.\mathbf{b}xv^?) \\
&\rightarrow_\beta \forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \rightarrow (\lambda x.\mathbf{b}xv^?)x) \\
&\rightarrow_\beta \forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \rightarrow \mathbf{b}xv^?) \qquad (2.15)
\end{aligned}$$

Interpretation (2.15), computed compositionally, has all the problems of cross-sentential anaphora discussed above. It is not clear how to link the free variable $v^?$ standing for the pronoun *it* to its antecedent. Firstly, the free variable is not informative to retrieve the antecedent. It is just a dummy variable that is used to fill the second argument of **b**. The semantical content of the pronoun *it* is lost, which happened already at the level of lexical interpretation. Indeed, the interpretations of other pronouns are exactly the same in this approach. Secondly, even if it were possible to retrieve the antecedent, which is interpreted by variable z in (2.15), the formula $\mathbf{b}xv^?$ is located outside of the scope of the existential quantifier that binds z, therefore substituting $v^?$ with z would not change anything from logical point of view. Moreover, in the desired interpretation, not only $\mathbf{b}xv^?$ should be within the scope of the quantifier binding z, but also the quantifier should be universal. It is an important problem to keep the assumption that an indefinite introduces an existential quantification, but have it changed to a universal quantifier under certain circumstances. This problem and others seem to be unsolvable within a static view of semantics.

2.1.4 Presuppositions

It has long been discussed, by proponents of pragmatic and semantic characterizations of presupposition, what presupposition is exactly. According to the former, (pragmatic) presupposition should be defined and explained relative to a context, including knowledge of the speaker and hearer and conversational principles. In contrast, according to the latter, (semantic) presupposition is a semantic relation between sentences (additional to the relation of entailment).[10] These so-called

[10]The term "semantic presupposition" can be misleading. So-called semantic theories of presupposition are usually theories that are concerned with the truth value of the presupposition as a prerequisite for the truth value of a sentence (and involving either partial or multivalent accounts to the problems of presupposition). One of the most simplistic definition is, for example, as follows: A sentence A semantically presupposes a sentence B iff $A \models B$ and $\neg A \models B$, where $\models$ is the logical relation of entailment.

This does not mean, however, that if a theory of formal semantics is dealing with presupposition, it necessary falls into the group of so-called semantic theories of presupposition.

semantic theories of presupposition have many issues and fail to explain various non-trivial presuppositional phenomena[11] that can be more easily explained with a pragmatic view on presupposition. This led to the recognition of the importance of integrating semantic and pragmatic views on presupposition and to the development of various extensions of the former with a flavour of the latter (e.g. [Seuren, 1985] and [Burton-Roberts, 1989]), culminating with the appearance of new and more sophisticated approaches [Karttunen, 1973; Gazdar, 1979; van der Sandt, 1988]. These approaches are, however, not genially compositional and they analyse presuppositions in a somewhat descriptive fashion.[12]

The strong influence of contextual factors on the behaviour of a presupposition led to consideration of presuppositions as "background beliefs of the speaker" [Stalnaker, 1974]:

> A proposition P is a pragmatic presupposition of a speaker in a given context just in case the speaker assumes or believes that P, assumes or believes that his addressee assumes or believes that P, and assumes or believes that his addressee recognizes that he is making these assumptions, or has these beliefs. [Stalnaker, 1974]

If a speaker utters (15a), the information that some Frenchman walks in the park becomes part of the knowledge (context) of the hearer.[13] Clearly, the speaker and the hearer are aware of that. Then, continuing with (15b), the speaker makes use of the fact that he/she is aware that the knowledge of existence of the Frenchman (under discussion) is in the context of the hearer. Thus, the hearer is capable of identifying the Frenchman from the second sentence with the Frenchman from the first sentence.

(15) a. *A Frenchman walks in the park.*

b. *The Frenchman whistles.*

In a similar way, if the speaker utters (15b) in isolation, she/he assumes that the hearer knows about the existence of the Frenchman (under discussion) and the hearer recognizes that the speaker assumes this. This forces the hearer to enlarge her or his context (knowledge base) with the presupposed proposition of the existence of the Frenchman.

[11] The limitations of so-called semantic accounts of presupposition were already recognized by Stalnaker [Stalnaker, 1973; Stalnaker, 1974]. See [van der Sandt, 1988] and [Beaver, 2001] for more technical discussions.

[12] Additional discussion on limitations of [Karttunen, 1973; Gazdar, 1979; van der Sandt, 1988] can be found in [Beaver, 2001].

[13] Stalnaker analysed presuppositions not with respect to the knowledge of the hearer, but with respect to a "presumed common knowledge" [Stalnaker, 1973] between the speaker and the hearer. Whether the context is considered to be the common ground or the representation of the knowledge of the hearer only, it is clear that both notions of context play a major role.

The action of accepting (e.g. enlarging the context with) the presupposed knowledge, as the hearer does when (15b) is uttered in isolation, is known as **presupposition accommodation**:

> If at time t something is said that requires presupposition P to be acceptable, and if P is not presupposed just before t, then - *ceteris paribus* and within certain limits - presupposition P comes into existence at t. [Lewis, 1979, p.340]

Presuppositions are normally associated with expressions involving certain lexical items. For example, the presupposition of the existence of the Frenchman above is caused, or **triggered**, by the definite description (involving the definite article). A presupposition triggering expression can be informally defined in the following way:

Definition 2.2. [Presupposition triggering expression] A **presupposition triggering expression** (**presuppositional expression**) is an expression the meaning of which either depends on the meaning of another expression contained in the context or, if a suitable candidate for this dependence is absent in the context, the descriptive content of the presupposition triggering expression is sufficiently reach for the discourse to be felicitous.

Various presupposition triggers have been identified.[14] Among them, for example, are referring expressions (definite descriptions, proper names, possessives), factive verbs, aspectual verbs, temporal clauses ... Thus, in (16)–(19) a-sentences (or, being pragmatically more careful, someone who utters them) presuppose that (at least) the corresponding propositions in b and c hold:

(16) a. *John saw the man with two heads.*

b. *There exists a person named John.*

c. *There exists a man with two heads.*

(17) a. *Martha regrets failing.*

b. *There exists a person named Martha.*

c. *Martha failed.*

(18) a. *Bob stopped playing football.*

b. *There exists a person named Bob.*

[14] A large list with examples can be found in [Levinson, 1983, p.181-185], which is itself a selection from an unpublished collection of Karttunen.

c. *Bob had been playing football.*

(19) a. *Sue cried before she finished her thesis.*

b. *There exists a person named Sue.*

c. *Sue finished her thesis.*

For getting more intuition about presuppositions, it worth to touch upon historical debates about referring expressions, which originated the interest to presuppositional phenomena.

Referring Expressions

Philosophical investigations of Frege about the nature of **referring expressions** in natural language are considered to be the origin of the debates related to presupposition [Levinson, 1983]. Frege wrote:

> If anything is asserted there is always an obvious presupposition that the simple or compound proper names used have reference. If one therefore asserts 'Kepler died in misery,' there is a presupposition that the name 'Kepler' designates something; but it does not follow that the sense of the sentence 'Kepler died in misery' contains the thought that the name 'Kepler' designates something. [Frege, 1952, p.69]

In more recent terminology, the presupposition of a proper name is that the proper name has a reference and the presupposition is *not* the part of an asserted (proffered) content of an expression that contains this proper name. Frege's distinction between sense and reference was an answer to one of the main concerns in philosophy at that time, particularly, how sentences like (20) could be understood when there is no actual king in France:

(20) *The King of France is bald.*

The sense of (20), for example, could be (2.16), where **bald** is a predicate and k is a constant:

$$\textbf{bald}\ k \tag{2.16}$$

The truth value of the logical representation of a sentence depends on the model in which it is interpreted.[15] Therefore, taking into account compositionality, the truth value of (2.16) depends on the interpretation of **bald** and k. The constant k represents the sense of *the King of France*. However, since *the King of France*

[15] Remember the discussion in Section 1.3, particularly Example 1.20.

cannot be interpreted in the present world, it has no reference in the present world. Therefore, Sentence (20) has sense (which can be logically represented as (2.16)), but it does not have a reference (a truth value) in the present world.[16]

A different answer was given by Russell. He insisted on a very particular logical formula (introducing an existentially quantified variable) for the meaning of referring expressions. Thus, according to Russell, Sentence (20) has the logical formula (2.17), where **king** is a property of being King of France and **bald** is a property of being bald:

$$\exists x.\mathbf{king}x \wedge \mathbf{bald}x \wedge (\forall y.\mathbf{king}y \;\rightarrow\; x = y) \tag{2.17}$$

Formula (2.17) can be paraphrased as "There is a King of France, who is bald, and there is no other King of France". Then, Sentence (20) is meaningful and it is interpreted as false in the present world (because there is no King in France in the present world).

Russell's analysis of referring expressions as a complex logical formula allowed him to explain why Sentence (21a) can be continued with (21b):

(21) a. *The King of France is not bald.*

b. *Because there is no King in France.*

Sentence (21a) has ambiguous meaning (w.r.t. the scope of the negation). It can mean (2.18), paraphrased as "There is a King of France, who is not bald, and there is no other King of France", which is is interpreted as false in the present world:

$$\exists x.\mathbf{king}x \wedge \neg\mathbf{bald}x \wedge (\forall y.\mathbf{king}y \;\rightarrow\; x = y) \tag{2.18}$$

Sentence (21a) can also mean (2.19), paraphrased as "It is not the case that (there is a King of France, who is bald, and there is no other King of France)", which is interpreted as true in the present world:

$$\neg(\exists x.\mathbf{king}x \wedge \mathbf{bald}x \wedge (\forall y.\mathbf{king}y \;\rightarrow\; x = y)) \tag{2.19}$$

Thus, the continuation (21b) is consistent with Sentence (21a) when it is interpreted as (2.19); but it is inconsistent with the non-negated sentence (with meaning (2.17)) and with (21a) interpreted as (2.18).

Note that in Frege's approach, the existence of a reference for a name (definite description) is a prerequisite for a sentence containing this name to have a truth value, but it is not a part of the sense of the sentence and it does not need to be satisfied. That is why Frege names this prerequisite "presupposition". Russell, in contrast, analyses a sentence containing a definite description as a logical formula with a sub-formula expressing the existence of the individual with a respective

[16] Recall Model 3 in the Example 1.20.

property of the description (as, for example, a property of being the King of France). Therefore, the existence of such an individual is part of the assertion of the sentence in Russell's approach (and the concept of presupposition is absent).

Russell's analysis of definite descriptions has been dominant until the emergence of criticism of Strawson [1950; 1952] who re-emphasized the importance of the notion to presupposition. Strawson claimed that what expressions and sentences mean is different from the use of expressions and sentences in a particular occasion (in a particular context):[17]

> To give the meaning of an expression (in the sense in which I am using the word) is to give *general directions* for its use to refer to or mention particular objects or persons; to give the meaning of a sentence is to give *general directions* for its use in making true or false assertions. It is not to talk about any particular occasion of the use of the sentence or expression. The meaning of an expression cannot be identified with the object it is used, on a particular occasion, to refer to. The meaning of a sentence cannot be identified with the assertion it is used, on a particular occasion, to make. For to talk about the meaning of an expression or sentence is not to talk about its use on a particular occasion, but about the rules, habits, conventions governing its correct use, on all occasions, to refer or to assert. [Strawson, 1950]

However, the reference of an expression or of a sentence is dependent on a particular occasion (context) in which it is used:

> The actual unique reference made, if any, is a matter of the particular use in the particular context; the significance of the expression used is the set of rules or conventions which permits such references to be made. [Strawson, 1950]

> What in general is required for making a unique reference is, obviously, some device, or devices, for showing both *that* a unique reference is intended and *what* unique reference it is; some device requiring and enabling the hearer or reader to identify what is being talked about. In securing this result, the context of utterance is of an importance which it is almost impossible to exaggerate; and by "context" I mean, at least, the time, the place, the situation, the identity of the speaker, the subjects which form the immediate focus of interest, and the personal histories of both speaker and those he is addressing. Besides context, there is, of course, convention; - linguistic convention. [Strawson, 1950]

[17] Therefore, Strawson's analysis can be seen as foregoing ideas for Kaplan's and Stalnaker's more formal analyses of the meanings of sentences as functions from contexts to propositions (see Subsection 2.1.1).

Returning to Sentence (20), Strawson considered that Russell was mistaken in assuming that anyone uttering it (as well as any other sentence) is necessarily making a true or a false assertion and that the present existence of one and only one King of France is a part of this assertion. According to him, (20) can be "seriously uttered" only when it is evident that there exists a King of France. On the other hand, if there is no present King in France, the question of whether the sentence is true or false "simply does not arise". Strawson called the relation between "certain subject-predicate statements" (e.g. (20)) and "certain existence-statements" (e.g. *There is a King of France*) the presupposition-relation. As Frege, Strawson considered the truth of the presupposition of a statement (i.e. a sentence uttered in a context) to be a necessary condition for the possibility of making an assertion (i.e. a necessary condition for the truth or falsity of the statement) [Strawson, 1952, p.175]. This shows that for Strawson, as for Frege, the presupposition is not part of the assertion.

Anaphora as Presupposition

Note that by Definitions 2.2 and 2.1, presupposition triggering expressions and anaphoric expressions are similar. Indeed, compare discourses in (13a) and in (15), repeated below in (22). Both the definite description *the Frenchman* in (22a) and the pronoun *he* in (22b) are anaphorically linked to the indefinite noun phrase *a Frenchman* from the first sentence.

(22) a. *A Frenchman walks in the park. The Frenchman whistles.*

b. *A Frenchman walks in the park. He whistles.*

For more examples of the analogy between presuppositional and anaphoric expressions, consider donkey sentences and compare a-sentences (with possesive noun phrases as presupposition triggering expressions) with b-sentences (with anaphoric pronouns) in (23) and (24).

(23) a. *If a farmer owns a donkey, he beats his donkey.*

b. *If a farmer owns a donkey, he beats it.*

(24) a. *Every farmer who owns a donkey beats his donkey.*

b. *Every farmer who owns a donkey beats it.*

The first theory based on the observation that the phenomena of presupposition and anaphora are analogous is developed under the claim that "presuppositional expressions are anaphoric expressions" [van der Sandt, 1992]. However,

as Geurts [1999] observes, it is contrariwise, i.e. anaphoric expressions are presuppositional expressions.[18] Therefore, to formally express the phenomena of presupposition, (at least) all questions related to formalization of anaphora, such as questions formulated in Subsections 2.1.2 and 2.1.3, should be answered.

For example, if the definite description (*the Frenchman*) in the second sentence in (15) is interpreted as a variable $y^{?}_{[the\ Frenchman]}$, where the superscript indicates that the anaphora should be attempted to be resolved and the subscript stores the descriptive content used in the case the antecedent for the anaphor is absent, then the logical meaning of Sentence (15b) can be represented as (2.20):

$$\textbf{whistles}\ y^{?}_{[the\ Frenchman]} \tag{2.20}$$

The composition of (2.20) with the logical formula (2.3) for the first sentence in (15) via the logical conjunction results in a formula having exactly the same issues as Formula (2.5):

$$(\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x) \wedge \textbf{whistles}\ y^{?}_{[the\ Frenchman]} \tag{2.21}$$

Projection Problem

In examples considered so far, presuppositions of expressions that trigger them were also presuppositions of the sentences containing these expressions. However, in some syntactic structures the presupposition triggered by the same expression can in some cases not **project** as a presupposition of the whole sentence. This phenomenon is called **presupposition projection problem** and happens, for example, for presuppositions triggered in indicative conditionals.

All four sentences in (25) contain a presupposition triggering noun phrase *his child*. It occurs in the consequent of the conditional in (25a), in the antecedent of the conditional in (25b), and in the consequent and in the antecedent in (25c). All of these sentences presuppose the existence of Tom's child. Sentence (25d) is different from the other three sentences in that it does not presuppose the existence of Tom's child, although it contains the presuppositional noun phrase in its consequent.

(25) a. *If Tom is back, his child is happy.*

b. *If Tom and his child go to swim, the water is warm.*

c. *If Tom praises his child, his child is happy.*

d. *If Tom has a child, his child is happy.*

[18] Stating it in terms of set theory, the set of anaphoric expressions is a subset of the set of presuppositional expressions.

The reason that the presupposition does not project in (25d) is that the antecedent of the conditional asserts what is presupposed in the consequent. Therefore, the content of the presupposition triggered by *his child* is already in the current context at the moment the antecedent is uttered.

When presupposition is not projected, there may be an impression that presupposition is cancelled or defeated, and indeed this terminology is widely used in the literature. Some theories even require presuppositions to be defeasible [Karttunen, 1973; Gazdar, 1979; van der Sandt, 1988]. However, if the presence of a context is acknowledged, it becomes clear that presupposition does not "disappear". There is always a presupposition caused by a presupposition triggering expression. This expression is evaluated with respect to the context at the moment of its utterance. Then, if the context already contains the content equivalent to the presupposed content (or entails it), the presupposition simply does not contribute to the knowledge in this context. If, however, there is no content in the context equivalent to the presupposition, the presupposition does contribute to the knowledge in the context. This goes in lines with Stalnaker's [1973; 1974] conception of presupposition.

Presupposition projection is a challenging problem for compositional semantics of natural language: the set of presuppositions of a complex expression is not necessarily the union of the sets of presuppositions of all its constituents. The answer to this problem requires a careful analysis of not only **how presuppositions are triggered**, but also **where they occur** and **how they are dependent upon the meanings of other expressions in the context.**

2.1.5 Conversational Implicatures

What is communicated in an utterance can often go beyond the meaning of what the utterance actually asserts. Consider an example from [Grice, 1975]:

A: Smith doesn't seem to have a girlfriend these days.
B: He has been paying a lot of visits to New York lately.

In a regular context, B communicates more to A than just informing about Smith frequently travelling to New York. What B means is that Smith has a girlfriend in New York and, therefore, he often goes to New York. Grice [1975] introduced a special term, **conversational implicature**, for the meaning that is not explicitly asserted but conveyed by an utterance in a particular context. Thus, B conversationally implicates that Smith has a girlfriend.

Understanding how conversational implicatures are triggered and computed are among the hardest open problems in discourse semantics. The difficulty is that the formal framework must include not only methods of compositionally computing meanings of complex expressions from conventional meanings of their constituents, but it also must contain some reasoning mechanism which would

compute conversational implicatures based on conventional meaning of an utterance and some background assumptions. To obtain a compositional formal framework with all these features is, however, not a trivial task.

2.2 Dynamic Approaches

This section presents fundamentals of the most influential dynamic approaches, particularly, Discourse Representation Theory, Dynamic Predicate Logic and Dynamic Montague Grammar.

2.2.1 Discourse Representation Theory

Discourse Representation Theory (DRT) was first presented in [Kamp, 1981] to study discourse and donkey anaphora.[19] It was introduced as a **representational theory**.

A sentence or a discourse is associated in DRT with a structure (called **discourse representation structure**, or DRS). A DRS is a box consisting of two parts: a set of **reference markers** (or **discourse referents**) and a set of **conditions**, as shown in (2.22):

$$\begin{array}{|c|}\hline x_1 \dots x_n \\ \hline C_1 \\ \vdots \\ C_m \\ \hline \end{array} \tag{2.22}$$

Formally, DRSs are the objects generated by the following grammar:

$$\begin{aligned} D &::= \{\{x_1, \dots, x_n\}, \{C_1, \dots, C_m\}\} \\ C &::= \top \mid Pt_1 \dots t_k \mid \neg D \mid x = t \\ t &::= x \mid c \end{aligned}$$

where $x_1, \dots, x_n$ and x are variables, P is a predicate and c is a constant.

For example, Sentence (26) is represented with the structure (2.23), which can also be shown graphically as (2.24), where x is a referent being introduced by the indefinite noun phrase and **man** x and **walks_in_the_park** x are two conditions.

(26) *A man walks in the park.*

[19] The theories existing at that moment were quite ad-hoc and could handle only restricted cases of pronominal anaphora and donkey sentences. See [Heim, 1982, ch.1] for an overview.

$$\{\{x\}, \{\textbf{man}\ x, \textbf{walks_in_the_park}\ x\}\} \tag{2.23}$$

x
man x
walks_in_the_park x

(2.24)

Importantly, the content of a DRS is not only a meaning representation of a sentence or a discourse, but also serves as the context. DRT does not make the distinction between content and context advocated by Stalnaker [1973; 1974]. This prevents DRT from expressing that presupposition of an expression is not a part of the assertive content of this expression.

Discourse in DRT is processed incrementally. The DRS of each newcoming sentence is **merged** with the DRS representing the current discourse.[20] The reference markers of the discourse DRS serve (under certain constraints discussed below) as anaphoric antecedents to pronouns and other anaphoric expressions. To avoid the scope problem discussed in Sections 2.1.2 and 2.1.3, the referents are considered as free variables from a technical point of view.

Sentence (27), having an anaphoric pronoun, is also represented with a DRS having a (free) variable, as shown in (2.25). Since this variable stands for a pronoun, it has to be bound to some discourse referent in the representation of the preceding discourse. To indicate that the variable awaits to be bound, the question mark superscript is used in (2.25):

(27) *He whistles.*

whistles $y^?$

(2.25)

Having representations of sentences (26) and (27), the representation of the discourse (12b), repeated in (28), can be computed:

(28) *A man walks in the park. He whistles.*

When the first sentence is uttered in an empty discourse, its representation (2.23) is merged with an empty DRS $\{\{\}, \{\}\}$, resulting in the representation of discourse after the utterance of (26), which is exactly like (2.23). Then, the DRS of the second sentence is merged with the current discourse DRS, resulting in the representation shown in (2.26):

x
man x
walks_in_the_park x
whistles $y^?$

(2.26)

[20] In the original definition of DRT, [Kamp, 1981; Kamp and Reyle, 1993], a newcoming sentence does not get an individual interpretation as a DRS, but is integrated directly into the discourse DRS.

Then, as the question mark indicates, the anaphora has to be resolved. DRT assumes that an anaphora resolution algorithm is given, and thus it is known that $y^?$ should be identified with x. Hence, DRS (2.26) is transformed into DRS (2.27), and the latter is equivalent to (2.28):

x
man x **walks_in_the_park** x **whistles** $y^?$ $y^? = x$

(2.27)

x
man x **walks_in_the_park** x **whistles** x

(2.28)

Each DRS can be translated to a formula in first order logic. While in DRS variables are considered as free variables, during the translation in first order logic they get an existential or a universal interpretation depending on the polarity of the DRS they belong to.[21] In the translation of (2.28), the variable x is interpreted existentially; and the whole DRS is translated into Formula (2.29), which is the desirable interpretation of (28):

$$\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x \wedge \textbf{whistles}\ x \tag{2.29}$$

In DRT the antecedent and a consequent of a conditional sentence are represented by separate DRSs connected with an arrow $\Longrightarrow$ that is analogous to the logical connective of implication. Indeed, the structure (2.30) is an abbreviation for (2.31):

$$\{\{x_1, \ldots, x_n\}, \{C_1, \ldots, C_m\}\} \Longrightarrow D \tag{2.30}$$

$$\neg\{\{x_1, \ldots, x_n\}, \{C_1, \ldots, C_m, \neg D\}\} \tag{2.31}$$

Moreover, conditions of the form $D_1 \Longrightarrow D_2$ take part in so called **accessibility constraints** for variables occurring within D_1 and D_2. These accessibility constraints play an important role in explaining cross sentential and donkey anaphoric dependences. Variables of D_2, for example, can only be bound to discourse referents occurring either in D_1 or in the structures that contain the condition $D_1 \Longrightarrow D_2$. Moreover, any discourse referent from D_2 is only accessible for variables from the same structure, i.e. D_2. In general, accessibility constraints are informally formulated as follows:[22] for a given variable x

[21] This hybrid status of variables causes the trouble of carefully choosing names of the new reference markers when constructing a new DRS.

[22] A more formal definition of accessibility constraints can be found in [van der Sandt, 1992].

> The set of available markers consists of the markers of the current structure, plus the markers of structures that can be reached from the current one by a series of steps in the directions *left*, (i.e. from the consequent of a pair $K \Longrightarrow K'$ to the antecedent), and *up*, (i.e. from a structure to an encompassing structure). [van Eijck and Kamp, 1997, p.187][van Eijck and Kamp, 2011]

Accessibility constraints is DRT's explanation why the discourse in (29a) is felicitous, while the discourse in (29b) is not.

(29) a. *A man walks in the park. He whistles. He is thinking about his holidays.*

b. *If a man walks in the park, he whistles. *He is thinking about his holidays.*

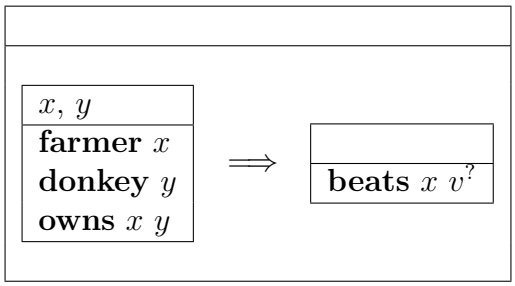
In the DRS for (29a) the reference marker for *a man* is contained in the universe of the main DRS and, therefore, is accessible to the variable interpreting the pronoun of the third sentence. However, in the DRS for (29b) the reference marker for *a man* is contained in the universe of discourse structure D_1 interpreting *if a man walks in the park*, which is a part of a complex condition $D_1 \Longrightarrow D_2$ (where D_2 is a structure interpreting *he whistles*). $D_1 \Longrightarrow D_2$ and the DRS interpreting the last sentence are both conditions of the main DRS, which makes the reference marker for *a man* not accessible for the variable interpreting the pronoun in the last sentence.

For an illustration how donkey sentences are analysed in terms of DRT's accessibility constraints, recall (14), repeated in (30).

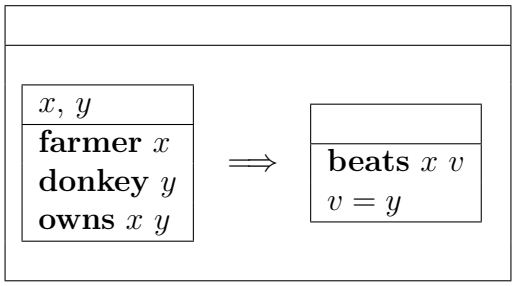
(30) a. *If a farmer owns a donkey, he beats it.*

b. *Every farmer who owns a donkey beats it.*

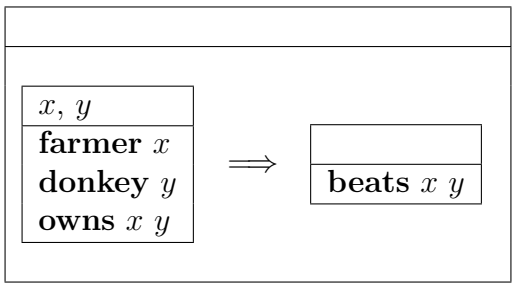
The DRS representing these sentences before anaphora is resolved is shown in (2.32). Variable $v^?$ needs to be bound and the necessary antecedent can only be found along the path of referents accessible for $v^?$. Therefore, x and y are the potential antecedents. Then, after assuming that anaphora is resolved and y is chosen, the representation becomes (2.33), which is equivalent to (2.34):



(2.32)



(2.33)

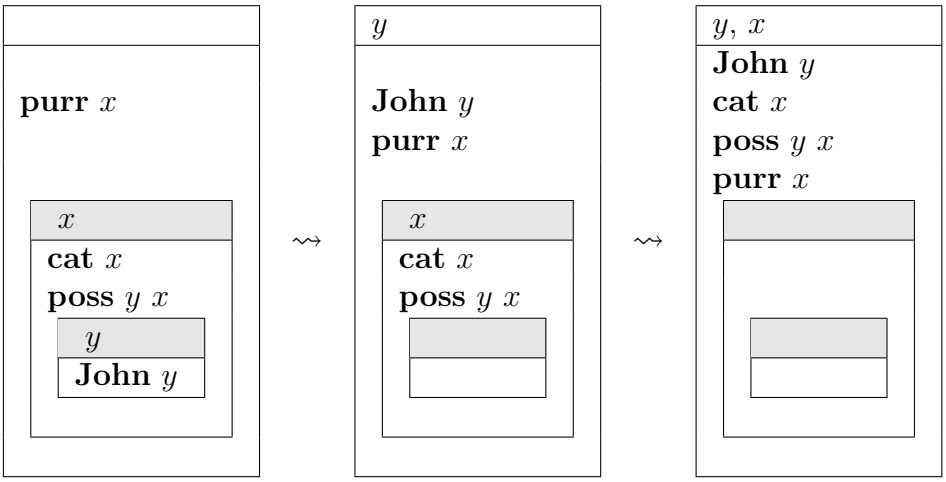


(2.34)

The clear and intuitive way of anaphora binding in DRT evolved into a powerful theory of presupposition projection [van der Sandt, 1992]. This theory treats presuppositions as anaphora with the only difference that their descriptive content can be accommodated (by establishing a reference marker) in the case there is no appropriate antecedent in the accessible DRS. Accessibility is a major constraint on the possibility of the anaphoric binding and, hence, it governs presupposition projection. Consider, for example, Sentence (31) from [van der Sandt, 1992]:

(31) *John's cat purrs.*

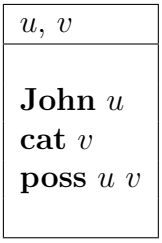
Initially the first structure below is associated with (31), where DRS with grey headings, called **A-structures**, indicate that they consist of anaphoric material, which is the possessive construction and the proper name. The latter is the constituent of the former and that is why its structure is contained in the structure for the possessive noun phrase. This DRS cannot be model-theoretically interpreted and has to be resolved, i.e. transformed to a DRS with an emply A-structure. First of all, the anaphor in the deepest A-structure has to be resolved. Since there is no suitable antecedent for it along the accessibility line, the anaphoric material of the proper name is accommodated in the top-most structure, resulting in the second DRS below. The anaphoric content of the remaining A-structure is accommodated in the top-most DRS as well, also due the absence of an antecedent for the anaphora. Thus, the final proper structure (that can be model-theoretically interpreted) for (31) is the third one shown below:



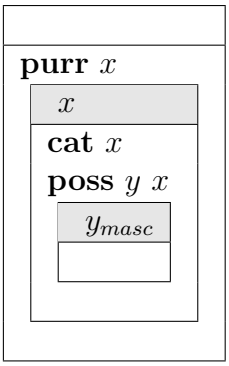
Consider now a tiny discourse (32) consisting of two sentences:

(32) *John has a cat. His cat purrs.*

The DRS for the first sentence in (32) is as follows:

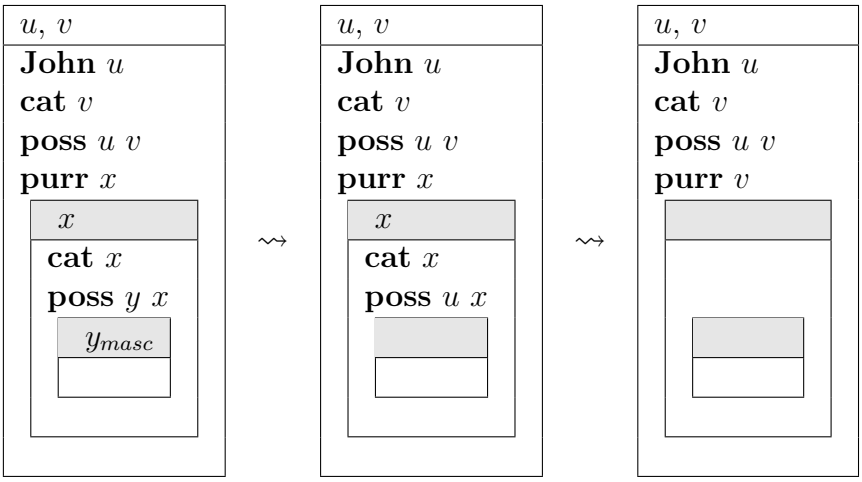


The DRS for the second sentence in (32) is shown immediately below. Note that it is almost like the first unresolved DRS for (31) with the only difference that the deep-most A-structure for *His cat purrs* contains a variable for pronoun (without descriptive material and hence cannot be independently resolved to a proper DRS) and the deep-most A-structure for *John's cat purrs* contains a variable for the proper name (with descriptive material that allowed it to be resolved on its own).



Merging the structures for the first and the second sentences in (32) leads to the first DRS shown below. It is resolved in the following way. First the variable y

in the deepest A-structure is equated with u leading to the second DRS below. Then the variable x in still unresolved A-structure is equated with v and the associated conditions are transferred to the top-most DRS, resulting in the third DRS shown below. This DRS is proper.



Note that the final proper structures obtained for (31) and (32) are equivalent. Hence, by van der Sandt's DRT account (31) and (32) are semantically equivalent.

Each DRS can be translated to a formula in first order logic, and vice versa. This is formally shown in [van Eijck and Kamp, 1997; van Eijck and Kamp, 2011]), and is also demonstrated below.

Translation $\ominus$ from DRT to FOL:

- $\{\{x_1, \ldots, x_n\}\{C_1, \ldots, C_m\}\}^{\ominus} \doteq \exists x_1, \ldots x_n . C_1^{\ominus} \wedge \cdots \wedge C_m^{\ominus}$
- $C^{\ominus} \doteq C$
- $(\neg C)^{\ominus} \doteq \neg C^{\ominus}$

Translation $\oslash$ from FOL to DRT:

- if Ψ is an atomic formula, $\Psi^{\oslash} \doteq \{\{\}, \{\Psi\}\}$
- $(\Phi \wedge \Psi)^{\oslash} \doteq \{\{\}, \{\Phi^{\oslash}, \Psi^{\oslash}\}\}$
- $(\neg\Psi)^{\oslash} \doteq \{\{\}, \neg\{\Psi^{\oslash}\}\}$
- $(\exists x_1 \ldots x_n . \Psi)^{\oslash} \doteq \{\{x_1, \ldots, x_n\}, \{\Psi^{\oslash}\}\}$, where Ψ does not begin with $\exists$

Since DRSs and formulas in first order logic can be mutually translated, a question arises why it is possible to assign the correct logical meaning for the discourses like (28) in DRT, but not in first order logic. This is explained in [van Eijck and Kamp, 1997] by "the different way in which DRT handles context". However, a thorough investigation reveals that DRSs are just a different notation.

Note that the first order logic formula (2.4) is equivalent to DRS (2.25) (remember $y^?$ is a free variable!). Moreover, note that interpretation (2.26), successfully built in DRT, is equivalent to (2.8), which cannot be constructed in first order logic. Why is it allowed to interpret the discourse as (2.26) in DRT, but not as equivalent one (2.8) in first order logic? How does DRT manage to place the content of the second sentence under the scope of the quantifier introduced in the first sentence? This is possible because in DRT's notation variables are not bound. However, it is not clear why it is advantageous to construct a representation with free variables, if a first order formula with free variables could be used instead. Then the interpretation of the first sentence of (28) would be (2.35) (compare it with DRS (2.24) and remember that x is free there), the interpretation of the second sentence of (28) would be (2.36) (compare it with DRS (2.25)) and the interpretation of the whole discourse, after the conjunction of the formulas for the first and for the second sentence and anaphoric linking, would be (2.37) (compare it with (2.27)) equivalent to (2.38) (compare it with (2.28)).

$$\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x \tag{2.35}$$

$$\textbf{whistles}\ y^? \tag{2.36}$$

$$\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x \wedge \textbf{whistles}\ y \wedge y = x \tag{2.37}$$

$$\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x \wedge \textbf{whistles}\ x \tag{2.38}$$

When (2.28) is translated into first order logic the variable x "becomes" existentially bound. Analogously, the variable x in (2.38) could be simply bound, which would result in (2.29), repeated in (2.39).

$$\exists x.\textbf{man}\ x \wedge \textbf{walks_in_the_park}\ x \wedge \textbf{whistles}\ x \tag{2.39}$$

A temporary neglection of quantifiers does not look natural when the first order logic notation is used. However, in the DRT's box notation the disregard of quantifiers is not so obvious and is masked under the definition of a DRS as a structure in which variables are not bound by a quantifier, and an assumption that a sentence or discourse is first of all interpreted in such a DRS.

It has been claimed that in its original definition [Kamp, 1981], and even in the later development [Kamp and Reyle, 1993], DRT is not compositional in the sense discussed in Section 1.2. Indeed, keeping in mind that DRSs are just a different notation for first order logic, it is clear that compositionality is not provided. Nevertheless, as demonstrated, for example, in [van Eijck and Kamp, 1997; van Eijck and Kamp, 2011], it is possible to add vocabulary and syntax of lambda calculus to the language of discourse representation structures. This amounts

not to an extension of DRT, but to creation of a new compositional theory using the language of discourse representation structures instead of a logical language. Insightful approaches concerned with extending DRT with compositionality include [Zeevat, 1989; Asher, 1993; Muskens, 1996; Brasoveanu, 2007].

In summary, DRT describes complex natural language phenomena using an intuitive graphical language, made of DRSs. DRT is therefore a convenient and useful framework for linguists to observe, analyse and report their work on complex natural language aspects such as anaphoric dependencies. However, DRT does not explain or compositionally solve the problems related to these phenomena. This still has to be done afterwards, preferably using classical and standard formalisms which are more suitable for implementation.

2.2.2 Dynamic Predicate Logic

Dynamic Predicate Logic (DPL) was developed in [Groenendijk and Stokhof, 1991] as an alternative to DRT. According to Groenendijk and Stokhof [1991], the main problem with DRT is that it does not satisfy the principle of compositionality and the reason is that DRT uses an intermediate level of semantic representation. The authors argue that an intermediate level also prevents a rich interpretation of natural language.

> There are several reasons why we think that the move to a semantic theory which assumes such an independent level of semantic representation, distinct both from syntactic structure and from meaning proper, should be looked upon with reserve. [Groenendijk and Stokhof, 1991, p.53].

An arbitrary intermediate level would, indeed, lead to an absurd theory. However, if the intermediate logical language is a translation of a natural language in accordance with the principle of compositionality, it is ungrounded to say that it is distinct from syntactic structure. Then, if the intermediate language is a language with well-studied model theoretical interpretation (as is the case of the language of DRT, since it can be easily translated to the usual language of first order logic), it is unfair to claim that it lacks a proper meaning. Moreover, as discussed in Section 1.4, a homomorphic interpretation a (natural) language into an intermediate logical language not only does not hurt, but is also useful.

DPL is claimed in [Groenendijk and Stokhof, 1991] to provide natural semantics without relying on such an intermediate level of semantic representation. However, this is an odd position, because the language of DPL clearly functions as an intermediate semantic representation for natural language.[23] Although in [Groenendijk and Stokhof, 1991] the model-theory of DPL is described, no

[23] Indeed, in [1990] Groenendijk and Stokhof seem to realize that the language of DPL is an intermediate language.

translation from natural language to language of DPL is ever defined. This is not a trivial task, and nevertheless it is simply silently avoided in DPL.

Another drawback of DRT, according to the authors of DPL, is that DRT uses a "non-orthodox" logical language. DPL is claimed to have an advantage over DRT in that it uses the language of first order logic. However, as discussed in Section 2.2.1 and as Groenendijk and Stokhof also observe, the language of DRT is equivalent to the language of first order logic, in the sense that translations between them can be defined. Therefore, it is not clear why DPL should be considered superior on these grounds. Furthermore, despite the authors' belief that it is undesirable to employ a non-standard apparatus (such as the language in DRT), they seem not to be disturbed by the development of highly non-standard model-theory within DPL.

It is important to emphasize that since a translation of natural language into the language of DPL is not provided, natural language phenomena are handled in the model-theoretical interpretation of the DPL language. Thus, in order to cope with cross-sentential anaphora, DPL interprets an existentially quantified formula and a conjunction of two formulas in a way that the formulas $(\exists x.Px) \wedge Qx$ and $\exists x.(Px \wedge Qx)$ get the same model-theoretical interpretation. Notice that the free variable in the second conjunct should be the same as the existentially quantified variable in the first conjunct. To handle donkey anaphora, DPL interprets implication of two formulas in a way that $(\exists x.Px) \rightarrow Qx$ and $\forall x.(Px \rightarrow Qx)$ also get the same model-theoretical interpretation. The free variable in the consequent has to be the same as the quantified variable in the antecedent of the conditional. Thus, natural language phenomena related to quantifier scope are handled in a rather ad-hoc way.

Turning to examples, consider once again the discourse (12b), repeated in (33):

(33) *A man walks in the park. He whistles.*

It can be represented with the DPL formula (2.40a).[24] Since (2.40a) is equivalent in DPL to (2.40b), the latter is taken to be the resulting meaning of (33). Therefore, "the occurrence of x in the last conjunct of (2) [(2.40a)], which is outside the scope of the existential quantifier, is still bound by that quantifier."[Groenendijk and Stokhof, 1990, p. 2]. This is rather conter-intuitive.

$$\exists x.(\mathbf{man}\ x \wedge \mathbf{walk_in_the_park}\ x) \wedge \mathbf{whistle}\ x \tag{2.40a}$$

$$\exists x.(\mathbf{man}\ x \wedge \mathbf{walk_in_the_park}\ x \wedge \mathbf{whistle}\ x) \tag{2.40b}$$

Regarding the DPL's account of donkey sentences, consider, for example, (34).

[24] Remember that the translation of natural language into DPL language is not defined in [Groenendijk and Stokhof, 1991]. Therefore, it is not clear how this formula, as well as any other formula in DPL, is computed.

(34) *Every farmer who owns a donkey beats it.*

It is represented in the language of DPL as (2.41a), which gets the same DPL model-theoretical interpretation as (2.41b). "In DPL an existential quantifier inside the antecedent of a conditional can bind free variables in the consequent, with the effect of universal quantification over the implication as a whole" [Groenendijk and Stokhof, 1990, p. 2]. This is also ad-hoc.

$$\forall x.(\textbf{farmer}\ x \wedge \exists y.(\textbf{donkey}\ y \wedge \textbf{own}\ x\ y)) \rightarrow \textbf{beat}\ x\ y \tag{2.41a}$$

$$\forall x \forall y.((\textbf{farmer}\ x \wedge \textbf{donkey}\ y \wedge \textbf{own}\ x\ y) \rightarrow \textbf{beat}\ x\ y) \tag{2.41b}$$

The limitation of the theory of DPL on the choice of the free variables in the second conjunct or in the consequent of the implication (which should be the same as those bound by the quantifier) means that the anaphora is assumed to be already resolved in the DPL formula. However, this results in two shortcomings. First, the DPL framework does not provide any room for integrating an anaphora resolution algorithm. Second, which is a consequence of the first, the names of the variables have to be carefully chosen to avoid destructive assignment when conjoining DPL formulas corresponding to several sentences.

2.2.3 Dynamic Montague Grammar

Dynamic Montague Grammar (DMG) is a framework designed in [Groenendijk and Stokhof, 1990]. As the original Montague semantics, it uses λ-terms to express the meaning of lexical items, and thus the meaning of a complex expression can be computed simply by application of the λ-terms representing the meanings of the lexical items that constitute the expression. While the original DRT and DPL need to reinvent composition, DMG reuses the notion of composition that already existed in lambda calculus. DMG can therefore be considered more parsimonious, and hence superior.

However, to deal with dynamic phenomena, DMG extends the simply-typed lambda calculus with many new features. In addition to the symbols of first order logic, DMG has symbols standing for so called **discourse markers** and syntactic constructions called **state switchers**. The discourse markers are special status variables that are used to interpret indefinites and pronouns. They are distinguished from regular variables because they are to be "switched" to a variable that is bound by a quantifier that should actually bind the discourse marker. The switching is done by a state switcher, which has the form $\{x/d\}$ and specifies for each bound variable x which discourse marker d it should substitute. State switchers are introduced in the term (2.42) standing for dynamic existential quantifier and are propagated deep inside the logical formula, as, for example, shown in (2.43) for conjunction:

$$\mathcal{E}d.\mathbf{A} \doteq \lambda p.\exists x.\{x/d\}\mathbf{A}p \tag{2.42}$$

$$\{x/d\}(A \wedge B) \text{ is equivalent with } \{x/d\}A \wedge \{x/d\}B \tag{2.43}$$

When a switcher reaches the corresponding discourse marker, the substitution takes place, as shown in (2.44):

$$\begin{aligned} &\{x/d\}d \text{ is equivalent with } x \\ &\{x/d\}d' \text{ is equivalent with } d' \text{ if } d' \neq d \end{aligned} \tag{2.44}$$

States in DMG are assignments of values to discourse markers which act as a **parameter** for interpretation in a model. State switchers restrict the interpretation to those states in which discourse markers have a value. Then the logical formula of each subsequent natural language sentence is model-theoretically interpreted with respect to the current state (which is based on the previously computed formulas).

The set T of types in DMG is defined as follows:

1. $e, t \in T$ (atomic types)
2. if $\alpha, \beta \in T$, then $(\alpha \rightarrow \beta) \in T$
3. if $\alpha \in T$, then $(s \rightarrow \alpha) \in T$

The third item introduces a particular type $(s \rightarrow \alpha)$, which is an intensional analog of type α.[25] An expression of type $(s \rightarrow \alpha)$ is a function from a set of states to the particular individual in a model of type α. The fact that s is not a type but participates in forming more complex types results in an odd type system.

The syntax of DMG consists of variables and constants of any type, formulas of type t closed under logical connectives and quantifiers (i.e. $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\exists$, $\forall$) in a standard way, the λ-abstraction, the λ-application and, additionally, discourse markers of type e and two operators $^\wedge$ and $^\vee$. The operators $^\wedge$ and $^\vee$ function in the following way: if ψ is of type α, then $^\wedge\psi$ denotes the intension of α, is of type $(s \rightarrow \alpha)$ and can be seen as an abstraction over a state; if φ is of type $(s \rightarrow \alpha)$, then $^\vee\varphi$ denotes the extension of φ, is of type α and can be seen as an application to a state.

A proposition A (of type t) is dynamized as shown in (2.45), where $\uparrow$ is a dynamization function, p is a variable of type $(s \rightarrow t)$ and $p \notin FV(\mathrm{A})$. Note that the operator $^\vee$ has to be applied to the variable p. This expresses an application of p to a current state.

$$\uparrow \mathrm{A} \doteq \lambda p.\mathrm{A} \wedge {}^\vee p \tag{2.45}$$

[25] DMG uses Montague's intensional logic notations and terminology. However, DMG disregards Montague's intensional parameters, i.e. worlds and times, and uses states instead (and for different reasons).

Moreover, a type-raised proposition $\mathbf{P}$ (of type $((s \to t) \to t)$) can be type-lowered by applying the function $\downarrow$ defined in Equation (2.46). Note that the constant $\top$ has to be intensionalized.

$$\downarrow \mathbf{P} \doteq \mathbf{P}(^{\wedge}\top) \tag{2.46}$$

Finally, since the state is implicit in DMG and switching back and forth between extensions and intensions is very frequent during the meaning computation, the resulting λ-terms contain numerous occurences of the operators $^{\wedge}$ and $^{\vee}$. They are, therefore, rather verbose, becomes more evident below.

Dynamic conjunction ; is defined in DMG as shown in (2.47), where $\mathbf{A}$ and $\mathbf{B}$ are terms of type $((s \to t) \to t)$ and p is a term of type $(s \to t)$. Dynamic conjunction amounts to the composition of terms (notice, however, that the result of application of $\mathbf{B}$ to p has to be intensionalized in order to be passed as an argument to the first conjunct $\mathbf{A}$):

$$\mathbf{A};\mathbf{B} \doteq \lambda p.\mathbf{A}(^{\wedge}(\mathbf{B}p)) \tag{2.47}$$

The definitions of dynamic existential quantifier (2.42) and conjunction (2.47) in a way that they expect a particular argument[26] result in a very important property of DMG for dealing with cross-sentential anaphora: the terms $(\mathcal{E}d.\uparrow \mathrm{A});\uparrow \mathrm{B}$ and $\mathcal{E}d.(\uparrow \mathrm{A};\uparrow \mathrm{B})$ β-reduce to the same normal form, and therefore are β-equivalent:

$$\begin{aligned}
(\mathcal{E}d.\uparrow \mathrm{A});\uparrow \mathrm{B} &= \lambda q.(\mathcal{E}d.(\uparrow \mathrm{A}))(^{\wedge}(\uparrow \mathrm{B}q)) && \text{(by (2.47))}\\
&= \lambda q.(\lambda p.\exists x.\{x/d\}(\uparrow \mathrm{A})p)(^{\wedge}((\uparrow \mathrm{B})q)) && \text{(by (2.42))}\\
&\to_\beta \lambda q.\exists x.\{x/d\}(\uparrow \mathrm{A})(^{\wedge}((\uparrow \mathrm{B})q))\\
&= \lambda q.\exists x.\{x/d\}(\lambda p.\mathrm{A} \wedge {}^{\vee}p)(^{\wedge}((\lambda p.\mathrm{B} \wedge {}^{\vee}p)q)) && \text{(by (2.45))}\\
&\to_\beta \lambda q.\exists x.\{x/d\}(\lambda p.\mathrm{A} \wedge {}^{\vee}p)(^{\wedge}(\mathrm{B} \wedge {}^{\vee}q))\\
&\to_\beta \lambda q.\exists x.\{x/d\}(\mathrm{A} \wedge {}^{\vee\wedge}(\mathrm{B} \wedge {}^{\vee}q))\\
&= \lambda q.\exists x.\{x/d\}(\mathrm{A} \wedge (\mathrm{B} \wedge {}^{\vee}q)) && (2.48)
\end{aligned}$$

$$\begin{aligned}
\mathcal{E}d.(\uparrow \mathrm{A};\uparrow \mathrm{B}) &= \mathcal{E}d.(\lambda q.(\uparrow \mathrm{A})(^{\wedge}((\uparrow \mathrm{B})q))) && \text{(by (2.47))}\\
&= \lambda p.\exists x.\{x/d\}(\lambda q.(\uparrow \mathrm{A})(^{\wedge}((\uparrow \mathrm{B}q))))p && \text{(by (2.42))}\\
&\to_\beta \lambda p.\exists x.\{x/d\}(\uparrow \mathrm{A})(^{\wedge}((\uparrow \mathrm{B})p))\\
&= \lambda p.\exists x.\{x/d\}(\lambda p.\mathrm{A} \wedge {}^{\vee}p)(^{\wedge}((\lambda p.\mathrm{B} \wedge {}^{\vee}p)p)) && \text{(by (2.45))}\\
&\to_\beta \lambda p.\exists x.\{x/d\}(\lambda p.\mathrm{A} \wedge {}^{\vee}p)(^{\wedge}(\mathrm{B} \wedge {}^{\vee}p))\\
&\to_\beta \lambda p.\exists x.\{x/d\}(\mathrm{A} \wedge {}^{\vee\wedge}(\mathrm{B} \wedge {}^{\vee}p))\\
&= \lambda p.\exists x.\{x/d\}(\mathrm{A} \wedge (\mathrm{B} \wedge {}^{\vee}p)) && (2.49)
\end{aligned}$$

[26] This argument can be see as a continuation of the term. Chapter 3 is devoted for introducing continuations.

Thus, due to equivalence of terms in (2.48) and (2.49), Equation (2.50) holds, where $\uparrow$A and $\uparrow$B are terms of type $((s \to t) \to t)$, p and q are terms of type $(s \to t)$:

$$(\mathcal{E}d.\uparrow \mathrm{A}); \uparrow \mathrm{B} =_\beta \mathcal{E}d.(\uparrow \mathrm{A}; \uparrow \mathrm{B}) \tag{2.50}$$

Taking into account that $\sim \mathcal{E}d.\mathbf{A} \equiv \mathcal{A}d. \sim \mathbf{A}d$, Equation (2.51) also holds in DMG:

$$(\mathcal{E}d.\mathbf{A}) \Rightarrow \mathbf{B} =_\beta \mathcal{A}d.(\mathbf{A} \Rightarrow \mathbf{B}) \tag{2.51}$$

where $\Rightarrow$ is implication, defined in (2.52a), $\mathcal{A}$ is universal quantifier, defined in (2.52b) and $\sim$ is negation, defined in (2.52c):

$$\mathbf{A} \Rightarrow \mathbf{B} \doteq \sim (\mathbf{A}; \sim \mathbf{B}) \tag{2.52a}$$

$$\mathcal{A}d.\mathbf{A} \doteq \lambda p.\forall x.\{x/d\}\mathbf{A}(^\wedge\top) \wedge^\vee p \tag{2.52b}$$

$$\sim \mathrm{A} \doteq \uparrow \neg \downarrow \mathrm{A} \tag{2.52c}$$

DMG assigns dynamic lambda terms to lexical expressions of natural language and computes meanings of sentences in a compositional fashion. However, it is not sufficient in DMG only to have interpretations of lexical items in order to compute the meaning of the whole sentence: the lexical items should be indexed in advance, and λ-terms are taken with the respective indexing. Consider, once again, the classical donkey sentence (35a):

(35) a. *Every farmer who owns a donkey beats it.*

b. *Every*$_1$ *farmer who owns* a_2 *donkey beats it*$_2$.

After assigning indexes in the sentence, (35b), the lexical items are interpreted as shown below, where x is a variable of type e; $\mathbf{f}$, $\mathbf{d}$ are constants of type $(e \to t)$; $\mathbf{o}$, $\mathbf{b}$ are constants of type $(e \to (e \to t))$; $\mathbf{P}$, $\mathbf{Q}$ are terms of type $(s \to (e \to ((s \to t) \to t)))$; $\mathbf{Y}$ is a term of type $(s \to ((s \to (e \to ((s \to t) \to t))) \to ((s \to t) \to t)))$ and d_1, d_2 are discourse markers:

$$\begin{aligned}
[\![farmer]\!] &= \lambda x.\uparrow \mathbf{f}x \\
[\![donkey]\!] &= \lambda x.\uparrow \mathbf{d}x \\
[\![owns]\!] &= \lambda \mathbf{Y}y.(^\vee\mathbf{Y}(^\wedge\lambda x.\uparrow \mathbf{o}xy)) \\
[\![beats]\!] &= \lambda \mathbf{Y}y.(^\vee\mathbf{Y}(^\wedge\lambda x.\uparrow \mathbf{b}xy)) \\
[\![a_2]\!] &= \lambda \mathbf{P}\mathbf{Q}.\mathcal{E}d_2.(^\vee\mathbf{P}d_2; ^\vee\mathbf{Q}d_2) \\
[\![every_1]\!] &= \lambda \mathbf{P}\mathbf{Q}.\mathcal{A}d_1.(^\vee\mathbf{P}d_1 \Rightarrow^\vee \mathbf{Q}d_1) \\
[\![it_2]\!] &= \lambda \mathbf{Q}.^\vee\mathbf{Q}d_2
\end{aligned}$$

These lexical interpretations cannot, in fact, be directly used for computing the meanings of more complex phrases. For example, the term $[\![a_2]\!]$ cannot be applied to the term $[\![donkey]\!]$ in order to compute the meaning of the noun phrase *a donkey*. The reason is that the first argument of $[\![a_2]\!]$ should be of type $(s \to (e \to ((s \to t) \to t)))$, while $[\![donkey]\!]$ is of type $(e \to ((s \to t) \to t))$. Therefore, $[\![donkey]\!]$ has to be intensionalized in order to be passed as an argument to $[\![a_2]\!]$. Thus, the meaning of *a donkey* is computed as $[\![a_2]\!](^[\![donkey]\!])$. This impossibility of direct functional application significantly complicates the compositional computation of meaning: the system has to be able to identify when and which terms have to be intensionalized.

There is an unnatural detail related to lexical interpretations of DMG as it is presented in [Groenendijk and Stokhof, 1990]. Type-raised interpretations of transitive verbs, such as *owns* and *beats*, do not receive two type-raised arguments, as is normally the case; and the interpretation of *who* is absent. Instead of giving a lexical interpretation for relative pronouns, DMG considers a relative clause to be a "syncategorematic construction", handled by the special rule (2.53)[27] shown below, where *np* is a common noun phrase followed by a restrictive clause *rc*:

$$[\![np\ rc]\!] = \lambda x.([\![np]\!]x; [\![rc]\!]x) \tag{2.53}$$

A possible reason why the authors had to introduce the "syncategorematic rule" (2.53) is that their lexical interpretation of verbs has unusual type and would lead to typing problems in compositional computation of a relative clause such as, for example, *who owns a donkey*. This is illustrated in detail below.

A relative pronoun is of the category $((np \to s) \to n \to n)$. Therefore, its interpretation should expect two type-raised arguments: the first argument corresponds to the interpretation of an expression of category $(np \to s)$, for example *owns a donkey*, and the second argument corresponds to the interpretation of an expression of category n, for example *farmer*. In DMG, the lexical interpretation of *who* can thus be defined as (2.54), where $\mathbf{R}$ is a variable of type $(s \to ((s \to (e \to ((s \to t) \to t))) \to ((s \to t) \to t))) \to ((s \to t) \to t)$, $\mathbf{Q}$ is a variable of type $(s \to ((s \to (e \to ((s \to t) \to t))) \to ((s \to t) \to t)))$, $\mathbf{P}$ is a variable of type $(s \to (e \to ((s \to t) \to t)))$:

$$[\![who]\!] = \lambda \mathbf{R}\mathbf{Q}x.(^\vee\mathbf{Q}x; ^\vee\mathbf{R}(^\wedge(\lambda \mathbf{P}.^\vee\mathbf{P}x))) \tag{2.54}$$

Focus on the first argument $\mathbf{R}$ of $[\![who]\!]$. In the case of the donkey sentence, it should be computed in DMG from the application of the interpretation of the transitive verb *owns* to the intensionalized interpretation of the noun phrase *a*

[27] The rule is presented as $[\![np\ rc]\!] = \lambda x.([\![np]\!]x; [\![rc]\!])$ in [Groenendijk and Stokhof, 1990]. However, the missing x in the second conjunct is probably a typo.

donkey. First, consider the computation of the meaning of a_2 *donkey*:

$$\begin{aligned}
[\![a_2]\!](^[\![donkey]\!]) &= (\lambda \mathbf{P}\mathbf{Q}.\mathcal{E}d_2.(^{\vee}\mathbf{P}d_2;^{\vee}\mathbf{Q}d_2))(^[\![donkey]\!]) \\
&\rightarrow_\beta \lambda \mathbf{Q}.\mathcal{E}d_2.(^{\vee}(^[\![donkey]\!])d_2;^{\vee}\mathbf{Q}d_2) \\
&= \lambda \mathbf{Q}.\mathcal{E}d_2.([\![donkey]\!]d_2;^{\vee}\mathbf{Q}d_2) \\
&= \lambda \mathbf{Q}.\mathcal{E}d_2.((\lambda x.\uparrow \mathbf{d}x)d_2;^{\vee}\mathbf{Q}d_2) \\
&\rightarrow_\beta \lambda \mathbf{Q}.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;^{\vee}\mathbf{Q}d_2) \qquad (2.55)
\end{aligned}$$

Then, compute the meaning of *owns* a_2 *donkey*:

$$\begin{aligned}
[\![own]\!](^([\![a_2]\!](^[\![donkey]\!]))) &= (\lambda \mathbf{Y}y.(^{\vee}\mathbf{Y}(^\lambda x.\uparrow \mathbf{o}xy)))(^([\![a_2]\!](^[\![donkey]\!]))) \\
&\rightarrow_\beta \lambda y.(^{\vee}(^([\![a_2]\!](^[\![donkey]\!])))(^\lambda x.\uparrow \mathbf{o}xy)) \\
&= \lambda y.([\![a_2]\!](^[\![donkey]\!])(^\lambda x.\uparrow \mathbf{o}xy)) \\
&= \lambda y.(\lambda \mathbf{Q}.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;^{\vee}\mathbf{Q}d_2))(^\lambda x.\uparrow \mathbf{o}xy)) \\
&\rightarrow_\beta \lambda y.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;^{\vee}(^\lambda x.\uparrow \mathbf{o}xy))d_2) \\
&= \lambda y.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;(\lambda x.\uparrow \mathbf{o}xy)d_2) \\
&\rightarrow_\beta \lambda y.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;\uparrow \mathbf{o}d_2y) \qquad (2.56)
\end{aligned}$$

Notice that, due to the fact that the meaning of transitive verb *own* is defined in [Groenendijk and Stokhof, 1990] in a non-standard way, the resulting term (2.56) is not type-raised, and, therefore, it cannot be used as an (intensionalized) argument for the interpretation of *who*. In order to fix it, the lexical interpretation of transitive verbs can be formulated in a standard fashion. In DMG this amounts to the λ-term shown in (2.57), where $\mathbf{X}$, $\mathbf{Y}$ are terms of type $(s \rightarrow ((s \rightarrow (e \rightarrow ((s \rightarrow t) \rightarrow t))) \rightarrow ((s \rightarrow t) \rightarrow t)))$; x, y are variables of type e and $\mathbf{o}$ is a constant of type $(e \rightarrow (e \rightarrow t))$.

$$[\![owns]\!] = \lambda \mathbf{Y}\mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.^{\vee}\mathbf{Y}(^\lambda y.\uparrow \mathbf{o}xy)) \qquad (2.57)$$

Then, after applying the interpretation (2.57) for *owns* to the intensionalized meaning (2.55) of a_2 *donkey*, the desirable interpretation (2.58) for *owns a donkey* can be computed:

$$\begin{aligned}
&[\![owns]\!](^([\![a_2]\!](^[\![donkey]\!]))) \\
&= (\lambda \mathbf{Y}\mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.^{\vee}\mathbf{Y}(^\lambda y.\uparrow \mathbf{o}xy)))(^([\![a_2]\!](^[\![donkey]\!]))) \\
&\rightarrow_\beta \lambda \mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.^{\vee}(^([\![a_2]\!](^[\![donkey]\!])))(^\lambda y.\uparrow \mathbf{o}xy)) \\
&= \lambda \mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.[\![a_2]\!](^[\![donkey]\!])(^\lambda y.\uparrow \mathbf{o}xy)) \\
&= \lambda \mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.(\lambda \mathbf{Q}.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;^{\vee}\mathbf{Q}d_2))(^\lambda y.\uparrow \mathbf{o}xy)) \\
&\rightarrow_\beta \lambda \mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;^{\vee}(^\lambda y.\uparrow \mathbf{o}xy)d_2)) \\
&= \lambda \mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;(\lambda y.\uparrow \mathbf{o}xy)d_2)) \\
&\rightarrow_\beta \lambda \mathbf{X}.^{\vee}\mathbf{X}(^\lambda x.\mathcal{E}d_2.(\uparrow \mathbf{d}d_2;\uparrow \mathbf{o}xd_2)) \qquad (2.58)
\end{aligned}$$

Now it is possible to compute the meaning for *who owns* a_2 *donkey* without having to apply the syncategorematic rule:

$$\begin{aligned}
&[\![who]\!](^\wedge([\![owns]\!](^\wedge([\![a_2]\!](^\wedge[\![donkey]\!])))))\\
&= (\lambda \mathbf{RQ}x.(^\vee\mathbf{Q}x;^\vee\mathbf{R}(^\wedge\lambda\mathbf{P}.^\vee\mathbf{P}x))))(^\wedge([\![owns]\!](^\wedge([\![a_2]\!](^\wedge[\![donkey]\!])))))\\
&\rightarrow_\beta \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;^\vee(^\wedge([\![owns]\!](^\wedge([\![a_2]\!](^\wedge[\![donkey]\!])))))(^\wedge\lambda\mathbf{P}.^\vee\mathbf{P}x)))\\
&= \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;([\![owns]\!](^\wedge([\![a_2]\!](^\wedge[\![donkey]\!]))))(^\wedge\lambda\mathbf{P}.^\vee\mathbf{P}x)))\\
&= \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;(\lambda\mathbf{X}.^\vee\mathbf{X}(^\wedge\lambda x.\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2)))(^\wedge\lambda\mathbf{P}.^\vee\mathbf{P}x)))\\
&\rightarrow_\beta \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;^\vee(^\wedge\lambda\mathbf{P}.^\vee\mathbf{P}x))(^\wedge\lambda x.\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2)))\\
&= \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;(\lambda\mathbf{P}.^\vee\mathbf{P}x)(^\wedge\lambda x.\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2)))\\
&\rightarrow_\beta \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;^\vee(^\wedge\lambda x.\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2))x)\\
&= \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;(\lambda x.\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2))x)\\
&\rightarrow_\beta \lambda\mathbf{Q}x.(^\vee\mathbf{Q}x;\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2))
\end{aligned} \tag{2.59}$$

In what follows the meaning of the computation in DMG of the meaning of the complete indexed sentence (35b) is demonstrated.

After applying the meaning (2.59) for *who owns* a_2 *donkey* to the intensionalized meaning of $farmer_1$ and performing β-reduction, the meaning for $farmer_1$ *who owns* a_2 *donkey* is as shown in (2.60):

$$\begin{aligned}
&([\![who]\!](^\wedge([\![owns]\!](^\wedge([\![a_2]\!](^\wedge[\![donkey]\!])))))))(^\wedge[\![farmer]\!])\\
&\rightarrow^*_\beta \lambda x.(\uparrow\mathbf{f}x;\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}xd_2))
\end{aligned} \tag{2.60}$$

Now the meaning of the indexed noun phrase $every_1$ *farmer who owns* a_2 *donkey* can be computed:

$$\begin{aligned}
&[\![every_1]\!](^\wedge(([\![who]\!](^\wedge([\![owns]\!](^\wedge([\![a_2]\!](^\wedge[\![donkey]\!])))))))(^\wedge[\![farmer]\!])))\\
&\rightarrow^*_\beta \lambda\mathbf{Q}.\mathcal{A}d_1.((\uparrow\mathbf{f}d_1;\mathcal{E}d_2.(\uparrow\mathbf{d}d_2;\uparrow\mathbf{o}d_1d_2))\Rightarrow^\vee\mathbf{Q}d_1)
\end{aligned} \tag{2.61}$$

Since *beats* is a transitive verb, instead of its [Groenendijk and Stokhof, 1990]'s lexical interpretation, here (2.62), analogous to the interpretation defined for *owns* above, is used in order to compute the meaning (2.63) of the indexed verb phrase *beats* it_2:

$$[\![beats]\!] = \lambda\mathbf{YX}.^\vee\mathbf{X}(^\wedge\lambda x.^\vee\mathbf{Y}(^\wedge\lambda y.\uparrow\mathbf{b}xy)) \tag{2.62}$$

$$[\![beats]\!](^\wedge[\![it_2]\!]) \rightarrow^*_\beta \lambda\mathbf{X}.^\vee\mathbf{X}(^\wedge\lambda x.\uparrow\mathbf{b}xd_2) \tag{2.63}$$

Finally, the meaning (2.64) of the complete indexed sentence is computed by applying the meaning of the verb phrase *beats* it_1 to the intensionalized meaning of the noun phrase *every* $farmer_1$ *who owns* a_2 *donkey*, β-reducing, propagating

the state switchers deep inside the formula and performing the substitutions of discourse markers to variables.[28]

$$
\begin{aligned}
& (\llbracket beats \rrbracket(\hat{}\llbracket it_2 \rrbracket))(\hat{}(\llbracket every_1 \rrbracket(\hat{}((\llbracket who \rrbracket(\hat{}(\llbracket owns \rrbracket(\hat{}(\llbracket a_2 \rrbracket(\hat{}\llbracket donkey \rrbracket)))))))(\hat{}\llbracket farmer \rrbracket)))) \\
\rightarrow^*_\beta\ & \mathcal{A}d_1.((\uparrow \mathbf{f}d_1; \mathcal{E}d_2.(\uparrow \mathbf{d}d_2; \uparrow \mathbf{o}d_1d_2)) \Rightarrow \uparrow \mathbf{b}d_1d_2) \\
=\ & \mathcal{A}d_1d_2.((\uparrow \mathbf{f}d_1; (\uparrow \mathbf{d}d_2; \uparrow \mathbf{o}d_1d_2)) \Rightarrow \uparrow \mathbf{b}d_1d_2) && \text{(by (2.51))} \\
\rightarrow^*_\beta\ & \mathcal{A}d_1d_2.\lambda p.((\mathbf{f}d_1 \wedge (\mathbf{d}d_2 \wedge \mathbf{o}d_1d_2)) \rightarrow \mathbf{b}d_1d_2) \wedge^\vee p && \text{(by def. of } \uparrow, ; \text{ and } \Rightarrow) \\
\rightarrow^*_\beta\ & \lambda p.\forall xy.\{x/d_1\}\{y/d_2\}((\mathbf{f}d_1 \wedge (\mathbf{d}d_2 \wedge \mathbf{o}d_1d_2)) \rightarrow \mathbf{b}d_1d_2) \wedge^\vee p && \text{(by (2.52b))} \\
=\ & \lambda p.\forall xy.((\mathbf{f}x \wedge (\mathbf{d}y \wedge \mathbf{o}xy)) \rightarrow \mathbf{b}xy) \wedge^\vee p && (2.64)
\end{aligned}
$$

The fact that DMG computes meanings in compositional fashion is a major advantage over other dynamic approaches, in particular DRT. On the other hand, the context is not explicit in DMG (in contrast to DRT). It is the state that functions as context in DMG because the meanings of expressions are state-dependent. For example, a referent in a model for a pronoun is determined by a state. The context dependence is implicitly encoded by the operation $^\vee$, i.e. via type lowering. The fact that the state is implicit restricts DRT from exploiting it in a more elaborate way, such as modifying it during the course of a discourse and reasoning on it, i.e. using it to account for pragmatical phenomena, such as presuppositions and conversational implicatures.

[28] In [Groenendijk and Stokhof, 1990], due to the unusual lexical interpretations of the transitive verbs, the interpretation of a noun phrase has to be applied to the intensionalized interpretation of the verb phrase in order to compute the meaning of the sentence. This is not very intuitive and is not in accordance with modern type logical grammars.

This chapter introduced non-trivial phenomena of natural language, such as, cross-sentential and donkey anaphora, presuppositions and conversational implicatures (Subsections 2.1.2– 2.1.5) that are considered to be dynamic due to their particular dependence on context (Subsection 2.1.1) and, therefore, have to be handled with a dynamic approach. Section 2.2 analysed the basics of the most prominent current dynamic frameworks, particularly Discourse Representation Theory, Dynamic Predicate Logic and Dynamic Montague Grammar. Although these frameworks represent context in different ways, they are similar in that they identify meaning with its context change potential. This is, however, not desirable. An expression can certainly have a context change potential, but it is not all it has.

Context change potential is part of the expression's meaning and could be modelled as a function that updates the context. For that, however, the meaning of an expression has to be first evaluated with respect to some context to actually cause the context change. Consequently, it is dependent on the hearer (or reader).

Additionally, an expression necessarily has an actual asserted content. This is the meaning of the expression that is independent of the hearer. Moreover, not all expressions are context dependent. A big part of natural language has rather standard meanings, which are independent of any context (but do cause changes in it). An example of such an expression is (36):

(36) *Every boy loves a girl.*

There seem to be no reason to handle this huge part of natural language as if it was context-dependent when it is actually not. Thus, it would be preferable to have a framework that has a clear distinction between what is asserted by an expression (or discourse) and what its context change potential is.

A framework acknowledging this distinction between content and context but at the same time implementing their possible interdependence would have a higher potential for handling pragmatic phenomena discussed in this chapter. Moreover, remembering that the ultimate technological goal is to parse a discourse and output a representation of this discourse that could be used by computers, the separation of the asserted content from the context change potential would allow computer to know the difference between what has actually been explicitly said and the implicit side effects that might occur for those who heard/read the discourse.

Chapters 4, 5, 6 and 7 are devoted to developing such a framework and showing how it can handle some of the pragmatical issues, particularly those associated with presuppositions triggered by referring expressions and factive verbs, presupposition projection, and some conversational implicatures. The framework extensively uses the notion of continuation from programming languages, which allows to define a dynamic compositional framework. The next chapter presents this notion in detail.

Chapter 3

Continuation

Montague's program requires the meaning of an expression to be computed compositionally from meanings of its lexical constituents. Dynamics of natural language makes, however, this task non-trivial, as discussed in the previous chapter. A similar kind of problem occurred in the mathematical semantics of programming languages, particularly in formalizing full jumps ("goto" statements) in a compositional way. To solve this problem, the method of continuations was introduced in [Strachey and Wadsworth, 1974] to extend the mathematical semantics of programming languages [Scott and Strachey, 1971] with a general formalized notion of control of full jumps. According to this method, if the evaluation of a program is in a state s, and the following command is c, continuation represents the state transition which would be produced up to the end of the program (i.e. the rest of the computation) after performing the state transformation specified by c. Every function of a program written in continuation-passing style (CPS) is given as an argument to the continuation of the program with respect to this function. Strachey and Wadsworth informally explained this in the following way:

> [...] our main semantic functions now take three arguments: an environment, a continuation and a store. [...] Together these arguments may be regarded as the semantic context in which each part of the program must be interpreted before its contribution to the meaning of the whole program can be determined. The environment and the store provide all necessary information about the history of the computation preceding the part under consideration; the continuation indicates how the computation will proceed to the end of the program unless the current control causes a jump. [Strachey and Wadsworth, 2000, p.147]

This chapter presents the basic principles of the continuation-passing technique and illustrates them on simple programming examples.

3.1 CPS Transformation

A program can be transformed from direct style to continuation-passing style (CPS). A possible way of CPS-transforming terms of typed lambda calculus was introduced in [Plotkin, 1975]. Plotkin's call-by-value transformation is shown in Definition 3.1:

Definition 3.1. [CPS-transformation] Let o be a distinguished type. For each term t of type ρ, it is possible to construct its CPS-transformation $\bar{t}$ of type $\overline{\rho}$ in the following way

$$\begin{aligned}
\overline{x} &= \lambda\phi^{\rho\to o}.\phi x^{\rho} \\
\overline{\lambda x^{\alpha}.N^{\beta}} &= \lambda\phi^{(\alpha\to\beta)'\to o}.\phi(\lambda x^{\alpha'}.\overline{N}^{\overline{\beta\to o}}) \\
\overline{M^{\alpha\to\beta}N^{\alpha}} &= \lambda\phi^{\beta'\to o}.\overline{M}^{\overline{\alpha\to\beta}}(\lambda m^{(\alpha\to\beta)'}.\overline{N}^{\overline{\alpha}}(\lambda n^{\alpha'}.mn\phi))
\end{aligned}$$

where ϕ of type $(\rho' \to o)$ is a **continuation** of t.

Each type $\overline{\rho}$ is defined as $(\rho' \to o) \to o$, where ρ' is as follows

$$\rho' = \begin{cases} \rho & \text{if } \rho \text{ is basic} \\ \alpha' \to (\beta' \to o) \to o & \text{if } \rho = \alpha \to \beta \end{cases}$$

For a better understanding of CPS-transformation, it is important to keep the type assignments in mind. That is why terms in Definition 3.1 are presented together with their types as superscripts. Equations (3.1) repeat equations in Definition 3.1 with types omitted for better readability:

$$\overline{x} = \lambda\phi.\phi x \tag{3.1a}$$

$$\overline{\lambda x.N} = \lambda\phi.\phi(\lambda x.\overline{N}) \tag{3.1b}$$

$$\overline{MN} = \lambda\phi.\overline{M}(\lambda m.\overline{N}(\lambda n.mn\phi)) \tag{3.1c}$$

It is possible to make a CPS-transformation of an application in a way that the argument is evaluated before the function, (3.2) shows the corresponding transformation rule:

$$\overline{MN} = \lambda\phi.\overline{N}(\lambda n.\overline{M}(\lambda m.mn\phi)) \tag{3.2}$$

The transformation presented above imposes call-by-value evaluation strategy in the resulting term. According to call-by-value strategy, the argument expression is evaluated before being passed to the function.[1]

[1] Call-by-value is usually contrasted with call-by-name evaluation strategy. According to call-by-name, the argument of a function is not evaluated before the evaluation of the function. Call-by-value evaluation is chosen here as it is closer to Montague's technique of type raising, as discussed in Section 3.3.

EXAMPLE 3.2. Consider a λ-term t computing $\sqrt{x^3}$:

$$t = \lambda x.\mathsf{sqrt}(\mathsf{cube}\ x)$$

where sqrt is a function computing the square root of its argument and cube is a function computing the cubic power of its argument. Term t can be CPS-transformed (to call-by-value evaluation style) in the following way:

$$\begin{aligned}
&\overline{\lambda x.\mathsf{sqrt}(\mathsf{cube}\ x)} \\
&= \lambda\phi.\phi(\lambda x.\overline{\mathsf{sqrt}(\mathsf{cube}\ x)}) && \text{(by (3.1b))} \\
&= \lambda\phi.\phi(\lambda x.(\lambda\phi'.\overline{\mathsf{sqrt}}(\lambda m.\overline{\mathsf{cube}\ x}(\lambda n.mn\phi')))) && \text{(by (3.1c))} \\
&= \lambda\phi.\phi(\lambda x.(\lambda\phi'.\overline{\mathsf{sqrt}}(\lambda m.(\lambda\phi''.\overline{\mathsf{cube}}(\lambda m'.\overline{x}(\lambda n'.m'n'\phi'')))(\lambda n.mn\phi')))) && \text{(by (3.1c))} \\
&\to^*_\beta \lambda\phi.\phi(\lambda x.(\lambda\phi'.\overline{\mathsf{sqrt}}(\lambda m.(\overline{\mathsf{cube}}(\lambda m'.\overline{x}(\lambda n'.m'n'(\lambda n.mn\phi'))))))) \\
&= \lambda\phi.\phi(\lambda x.(\lambda\phi'.\overline{\mathsf{sqrt}}(\lambda m.(\overline{\mathsf{cube}}(\lambda m'.(\lambda\phi''.\phi'' x)(\lambda n'.m'n'(\lambda n.mn\phi'))))))) && \text{(by (3.1a))} \\
&\to^*_\beta \lambda\phi.\phi(\lambda x.(\lambda\phi'.\overline{\mathsf{sqrt}}(\lambda m.(\overline{\mathsf{cube}}(\lambda m'.m'x(\lambda n.mn\phi')))))) && (3.3)
\end{aligned}$$

The CPS versions of sqrt and cube are shown in Equations (3.4a) and (3.4b) respectively:

$$\overline{\mathsf{sqrt}} = \lambda\phi.\phi(\lambda x\phi'.\phi'(\mathsf{sqrt}x)) \tag{3.4a}$$

$$\overline{\mathsf{cube}} = \lambda\phi.\phi(\lambda x\phi'.\phi'(\mathsf{cube}x)) \tag{3.4b}$$

After substituting $\overline{\mathsf{sqrt}}$ and $\overline{\mathsf{cube}}$ in (3.3) for (3.4a) and (3.4b) and β-reducing, the normalized CPS-transformed term shown below is obtained:

$$\bar{t} = \lambda\phi.\phi(\lambda x\phi'.\phi'(\mathsf{sqrt}(\mathsf{cube}x)))$$

□

Definition 3.1 transforms a program in direct style into an equivalent program in CPS. For example, t and $\bar{t}$ return the same result for the same argument 4, as Equations (3.5) and (3.6) in the next example show:

EXAMPLE 3.3.

$$\begin{aligned}
t(4) &= (\lambda x.\mathsf{sqrt}(\mathsf{cube}\ x))(4) \\
&\to_\beta \mathsf{sqrt}(\mathsf{cube}\ 4) \\
&= \mathsf{sqrt}\ 64 \\
&= 8 && (3.5)
\end{aligned}$$

$$\begin{aligned}\lambda\phi.\bar{t}(\lambda\phi'.\phi' 4\phi) &= \lambda\phi.(\lambda\phi.\phi(\lambda x\phi'.\phi'(\mathsf{sqrt}(\mathsf{cube}x))))(\lambda\phi'.\phi' 4\phi)\\ &\to_\beta \lambda\phi.(\lambda\phi'.\phi' 4\phi)(\lambda x\phi'.\phi'(\mathsf{sqrt}(\mathsf{cube}x)))\\ &\to_\beta \lambda\phi.(\lambda x\phi'.\phi'(\mathsf{sqrt}(\mathsf{cube}x)))4\phi\\ &\to_\beta \lambda\phi.\phi(\mathsf{sqrt}(\mathsf{cube}4))\\ &= \lambda\phi.\phi(\mathsf{sqrt}64)\\ &= \lambda\phi.\phi 8 \end{aligned} \tag{3.6}$$

□

3.2 CPS Control

As demonstrated in the previous section, by simply applying Definition 3.1 one transforms a program into an equivalent CPS program. However, the advantage of a program written in continuation passing style is that it can be expanded with unusual expressions managing its execution, like non-local transfers of control. For example, one of the possible control expressions of a function can be to discard all the future of the computation (i.e. its continuation) and return the error as a result of the whole program, in case an error happens within this function. Examples 3.4– 3.8 show how it can be done in the case of division by zero.

Example 3.4. Consider a λ-term t computing $\sqrt{\frac{x}{y}}$:

$$t = \lambda xy.\mathsf{sqrt}(\mathsf{div}\ x\ y) \tag{3.7}$$

During the evaluation of t, if the execution of div leads to an error, the error message is passed to the function sqrt as the argument. Assuming that sqrt is defined simply as a square root of its argument, this leads to sqrt being applied to a value for which it is not defined. For example, this happens if the second of the arguments given to t is zero:

$$\begin{aligned} t(9)(0) &= (\lambda xy.\mathsf{sqrt}(\mathsf{div}\ x\ y))(9)(0)\\ &\to_\beta (\lambda y.\mathsf{sqrt}(\mathsf{div}\ 9\ y))(0)\\ &\to_\beta \mathsf{sqrt}(\mathsf{div}\ 9\ 0)\\ &= \mathsf{sqrt}(\texttt{error})\\ &= ??? \end{aligned} \tag{3.8}$$

□

This situation could be handled in the original direct program by adding at the beginning of sqrt a conditional expression that checks whether the argument is an `error` (and returning an `error` in this case) or a number (and calculating

the square root of the number in this case). However, this means that, following this style, such a condition would have to be added to all other possible functions of the program. In contrast, the CPS technique allows to provide control to the division function itself in a way that it terminates the whole program when an error occurs during its execution.

It is possible to CPS-transform t according to Definition 3.1 and the next example demonstrates this. The transformations of functions sqrt and div, shown below, are necessary for intermediary steps:

$$\overline{\mathsf{sqrt}} = \lambda\phi.\phi(\lambda x\phi'.\phi'(\mathsf{sqrt}x)) \tag{3.9a}$$

$$\overline{\mathsf{div}} = \lambda\phi.\phi(\lambda x\phi'.\phi'(\lambda y\phi''.\phi''(\mathsf{div}\ x\ y))) \tag{3.9b}$$

Example 3.5.

$$\begin{aligned}
&\overline{\lambda xy.\mathsf{sqrt}(\mathsf{div}\ x\ y)} \\
&= \lambda\phi.\phi(\lambda x.(\overline{\lambda y.\mathsf{sqrt}(\mathsf{div}\ x\ y)})) && \text{(by (3.1b))} \\
&= \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.\overline{\mathsf{sqrt}(\mathsf{div}\ x\ y)}))) && \text{(by (3.1b))} \\
&= \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\overline{\mathsf{sqrt}}(\lambda m.\overline{\mathsf{div}\ x\ y}(\lambda n.mn\phi'')))))) && \text{(by (3.1c))} \\
&= \ldots && \text{(by (3.1c))} \\
&\to^*_\beta \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\overline{\mathsf{sqrt}}(\lambda m.\overline{\mathsf{div}}(\lambda m''.m''x(\lambda m'.m'y(\lambda n.mn\phi''))))))))) && (3.10) \\
&= \ldots && \text{(by (3.9a) and (3.9b))} \\
&\to^*_\beta \lambda\phi.\phi(\lambda x\phi'.\phi'(\lambda y\phi''.\phi''(\mathsf{sqrt}(\mathsf{div}\ x\ y)))) && (3.11)
\end{aligned}$$

□

Since normalized term $\bar{t}$ in (3.11) is obtained from t in (3.7) only by applying CPS-transformation definitions, it is equivalent to t and behaves analogously when an error happens inside the division function. To demonstrate it, $\bar{t}$ is "fed" in the next example with the same (continuized) arguments 9 and 0 as t in Example 3.4. The fact that the result is equally undesirable can be seen by comparing Equations (3.8) and (3.12):

Example 3.6.

$$\begin{aligned}
&\lambda\phi'''.\bar{t}(\lambda\phi''''.\phi''''9(\lambda\phi''''.\phi''''0\phi''')) \\
&= \lambda\phi'''.(\lambda\phi.\phi(\lambda x\phi'.\phi'(\lambda y\phi''.\phi''(\mathsf{sqrt}(\mathsf{div}\ x\ y)))))(\lambda\phi''''.\phi''''9(\lambda\phi''''.\phi''''0\phi''')) \\
&\to_\beta \lambda\phi'''.(\lambda\phi''''.\phi''''9(\lambda\phi''''.\phi''''0\phi'''))(\lambda x\phi'.\phi'(\lambda y\phi''.\phi''(\mathsf{sqrt}(\mathsf{div}\ x\ y)))) \\
&\to_\beta \lambda\phi'''.(\lambda x\phi'.\phi'(\lambda y\phi''.\phi''(\mathsf{sqrt}(\mathsf{div}\ x\ y))))9(\lambda\phi''''.\phi''''0\phi''') \\
&\to^*_\beta \lambda\phi'''.(\lambda\phi''''.\phi''''0\phi''')(\lambda y\phi''.\phi''(\mathsf{sqrt}(\mathsf{div}\ 9\ y))) \\
&\to_\beta \lambda\phi'''.(\lambda y\phi''.\phi''(\mathsf{sqrt}(\mathsf{div}\ 9\ y)))0\phi''' \\
&\to^*_\beta \lambda\phi'''.\phi'''\mathsf{sqrt}(\mathsf{div}\ 9\ 0)
\end{aligned}$$

$$= \lambda\phi'''.\phi'''(\mathsf{sqrt}(\mathtt{error})) \tag{3.12}$$
$$= ???$$

□

However, an advantage of the CPS-written program (3.11) is that it can be extended with an unusual control due to the fact that it, as well as its functions, have continuations as arguments. Particularly, division can be refined in a way so that when the divisor is zero, it disregards the continuation and immediately returns an error message as a result of the whole program; and otherwise provides the result of the division to its continuation. Thus, CPS-transformation of div can be done not directly by Equation (3.9b), but by using its modified version (3.13) with a control on whether y is equal to zero or not:

$$\overline{\mathsf{div}} = \lambda\phi.\phi(\lambda x\phi'.\phi'(\lambda y.\mathsf{if}\ (y = 0)\ \mathsf{then}\ \mathtt{error}\ \mathsf{else}\ (\lambda\phi''.\phi''(\mathsf{div}\ x\ y)))) \tag{3.13}$$

Then, using Equation (3.13) in (3.10), and following Definition 3.1 in the rest of the transformation, a new CPS program that has a special treatment for division by zero is obtained, as shown below:

Example 3.7.

$$\begin{aligned}
&\overline{\lambda xy.\mathsf{sqrt}(\mathsf{div}\ x\ y)}\\
=&\ldots && \text{(by (3.1b) and (3.1c))}\\
\to^*_\beta&\ \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\overline{\mathsf{sqrt}}(\lambda m.\overline{\mathsf{div}}(\lambda m''.m''x(\lambda m'.m'y(\lambda n.mn\phi''))))))))\\
=&\lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.(\lambda\phi.\phi(\lambda x\phi'.\phi'(\mathsf{sqrt}x)))) && \text{(by (3.9a))}\\
&&(\lambda m.\overline{\mathsf{div}}(\lambda m''.m''x(\lambda m'.m'y(\lambda n.mn\phi''))))))))\\
\to^*_\beta&\ \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\overline{\mathsf{div}}(\lambda m''.m''x(\lambda m'.m'y(\lambda n.\phi''(\mathsf{sqrt}n))))))))\\
=&\ \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.(\lambda\phi.\phi(\lambda x'\phi'.\phi'(\lambda y'.\mathsf{if}\ (y' = 0)\ \mathsf{then}\ \mathtt{error}\\
&&\mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x'\ y')))))\\
&&(\lambda m''.m''x(\lambda m'.m'y(\lambda n.\phi''(\mathsf{sqrt}n))))))))\\
&&& \text{(by (3.13))}\\
\to_\beta&\ \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.(\lambda m''.m''x(\lambda m'.m'y(\lambda n.\phi''(\mathsf{sqrt}n))))\\
&&(\lambda x'\phi'.\phi'(\lambda y'.\mathsf{if}\ (y' = 0)\ \mathsf{then}\ \mathtt{error}\\
&&\mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x'\ y'))))))))\\
\to_\beta&\ \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.(\lambda x'\phi'.\phi'(\lambda y'.\mathsf{if}\ (y' = 0)\ \mathsf{then}\ \mathtt{error}\\
&&\mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x'\ y'))))\\
&&x(\lambda m'.m'y(\lambda n.\phi''(\mathsf{sqrt}n)))))))\\
\to^*_\beta&\ \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.(\lambda m'.m'y(\lambda n.\phi''(\mathsf{sqrt}n))))\\
&&(\lambda y'.\mathsf{if}\ (y' = 0)\ \mathsf{then}\ \mathtt{error}\\
&&\mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x\ y')))))))
\end{aligned}$$

$$\begin{aligned}
&\to_\beta \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.(\lambda y'.\mathsf{if}\ (y'=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x\ y')))y(\lambda n.\phi''(\mathsf{sqrt}n))))))\\
&\to_\beta \lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x\ y))(\lambda n.\phi''(\mathsf{sqrt}n))))))
\end{aligned} \tag{3.14}$$

□

Term (3.14), abbreviated below as $\bar{t}^{control}$, computes a square root of a result of a division, as programs t and $\bar{t}$ do. However, $\bar{t}^{control}$ additionally has the control over division by zero. Particularly, when the divisor is equal to zero, $\bar{t}^{control}$, unlike t and $\bar{t}$, terminates with an error. This is demonstrated in Example 3.8:

Example 3.8.

$$\begin{aligned}
&\lambda\phi.\bar{t}^{control}(\lambda\phi'.\phi'9(\lambda\phi''.\phi''0\phi))\\
&= \lambda\phi.(\lambda\phi.\phi(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x\ y))(\lambda n.\phi''(\mathsf{sqrt}n)))))))\\
&\qquad(\lambda\phi'.\phi'9(\lambda\phi''.\phi''0\phi))\\
&\to_\beta \lambda\phi.(\lambda\phi'.\phi'9(\lambda\phi''.\phi''0\phi))\\
&\qquad(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x\ y))(\lambda n.\phi''(\mathsf{sqrt}n))))))\\
&\to_\beta \lambda\phi.(\lambda x.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ x\ y))(\lambda n.\phi''(\mathsf{sqrt}n))))))9(\lambda\phi''.\phi''0\phi)\\
&\to_\beta \lambda\phi.(\lambda\phi'.\phi'(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ 9\ y))(\lambda n.\phi''(\mathsf{sqrt}n)))))(\lambda\phi''.\phi''0\phi)\\
&\to_\beta \lambda\phi.(\lambda\phi''.\phi''0\phi)(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\\
&\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ 9\ y))(\lambda n.\phi''(\mathsf{sqrt}n))))\\
&\to_\beta \lambda\phi.(\lambda y.(\lambda\phi''.\mathsf{if}\ (y=0)\ \mathsf{then}\ \mathtt{error}\ \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ 9\ y))(\lambda n.\phi''(\mathsf{sqrt}n))))0\phi\\
&\to_\beta^* \lambda\phi.\mathsf{if}\ (0=0)\ \mathsf{then}\ \mathtt{error}\ \mathsf{else}\ (\lambda\phi'''.\phi'''(\mathsf{div}\ 9\ 0))(\lambda n.\phi(\mathsf{sqrt}n))\\
&= \mathtt{error}
\end{aligned}$$

□

As already mentioned above, the method of continuation was introduced to give compositional semantics to full jumps in programming languages. Next example illustrates on the example of a simple language that continuation-based interpretation of abort operation leads to desired evaluation of a program, while a direct interpretation does not:

Example 3.9. Consider a small programming language L_1 consisting of variables from set var, expressions from set expr, symbols := and ;. The language has the

following simple grammar, where S stands for a statement:

$$\begin{aligned} S \doteq\ & \mathsf{var} := \mathsf{expr} \mid \\ & S; S \end{aligned}$$

Let state be a function mapping variables to natural numbers:

$$\mathsf{state} = \ \mathsf{var} \to \mathbb{N}$$

The statements S in L_1 are interpreted with the function I that maps expressions to natural numbers and statements to transformation on states:

$$\begin{aligned} I_{\mathsf{expr}} &: \mathsf{expr} \to \mathbb{N} \\ I_S &: S \to (\mathsf{state} \to \mathsf{state}) \end{aligned}$$

Then, compositional semantics of this language can be defined as below, where ξ is a state:

$$\begin{aligned} I_S(x := e)\xi &= \xi[x := I_{\mathsf{expr}}(e)] \\ I_S(S_1; S_2)\xi &= I_S(S_2)(I_S(S_1)\xi) \end{aligned}$$

Consider, for example, the following program written in L_1:

$$\begin{aligned} & x := 3; \\ & x := 5 \end{aligned}$$

After the program is evaluated, the final state stores the value 5 for x:

$$\begin{aligned} I_S(x := 3; x := 5)\xi &= I_S(x := 5)(I_S(x := 3)\xi) \\ &= I_S(x := 5)\xi[x := 3] \\ &= \xi[x := 5] \end{aligned}$$

Assume now that the language L_1 is extended with an abort operation, resulting in new language L_2:

$$\begin{aligned} S \doteq\ & \mathsf{var} := \mathsf{expr} \mid \\ & S; S \mid \\ & \mathsf{abort} \end{aligned}$$

Intuitively, the interpretation of abort in a state ξ should return the same state:

$$I_S(\mathsf{abort})\xi = \xi$$

It is desirable that the following program, after its termination (due to abort) should store 3 as the value of x:

$$\begin{array}{l} x := 3; \\ \mathsf{abort}; \\ x := 5 \end{array}$$

However, evaluating the program according to the rules defined above, results in x having the value 5:

$$\begin{aligned} I_S(x := 3; \mathsf{abort}; x := 5)\xi &= I_S(x := 5)(I_S(x := 3; \mathsf{abort})\xi) \\ &= I_S(x := 5)(I_S(\mathsf{abort})(I_S(x := 3)\xi)) \\ &= I_S(x := 5)(I_S(\mathsf{abort})\xi[x := 3]) \\ &= I_S(x := 5)\xi[x := 3] \\ &= \xi[x := 5] \end{aligned}$$

This can be solved by defining compositional semantics using continuations. Then interpretation I_S is of the type $(S \to \mathsf{state} \to (\mathsf{state} \to \mathsf{state}))$ and the semantic rules are as follows:

$$\begin{aligned} I_S(x := e)\xi\phi &= \phi(\xi[x := I_{\mathsf{expr}}(e)]) \\ I_S(S_1; S_2)\xi\phi &= I_S(S_1)\xi(\lambda s. I_S(S_2)s\phi) \\ I_S(\mathsf{abort})\xi\phi &= \xi \end{aligned}$$

Evaluation of the program according to continuation-based interpetation rules results in x being assigned the desired value 3:

$$\begin{aligned} I_S(x := 3; \mathsf{abort}; x := 5)\xi\phi &= I_S(x := 3)\xi(\lambda s. I_S(\mathsf{abort})s(\lambda s'. I_S(x := 5)\xi\phi)) \\ &= I_S(x := 3)\xi(\lambda s. s) \\ &= \lambda s. s(\xi[x := 3]) \\ &= \xi[x := 3] \end{aligned}$$

When abort is absent, continuation-based evaluation of the program correctly assigns the value 5 to x:

$$\begin{aligned} I_S(x := 3; x := 5)\xi\phi &= \phi(\xi[x := 3][x := 5]) \\ &= \phi(\xi[x := 5]) \end{aligned}$$

□

3.3 Continuation Technique in Natural Language Semantics

Continuation-like approaches can already be found in earlier work on formal compositional semantics of natural language. Montague's [1973] technique of type-raising[2] can be seen as using continuation technique for quantifiers to take scope over complete sentences. In this type-raised setting, scope ambiguities (subject wide scope versus object wide scope) correspond to the change of evaluation order of the arguments. This is one of the features that continuation-style allows. Indeed, compare Montague's account of scope ambiguity shown in (1.7a) and (1.7b) and repeated below in (3.15), with respectively equations (3.1c) and (3.2) of call-by-value continuation-passing-style transformation, repeated below in (3.16), that impose different evaluation order of the arguments:

$$[\![loves]\!]_{sws} = \lambda OS.S(\lambda x.O(\lambda y.\mathbf{love}xy)) \tag{3.15a}$$
$$[\![loves]\!]_{ows} = \lambda OS.O(\lambda y.S(\lambda x.\mathbf{love}xy)) \tag{3.15b}$$

$$\overline{MN} = \lambda\phi.\overline{M}(\lambda m.\overline{N}(\lambda n.mn\phi)) \tag{3.16a}$$
$$\overline{MN} = \lambda\phi.\overline{N}(\lambda n.\overline{M}(\lambda m.mn\phi)) \tag{3.16b}$$

More recent research confirms that continuation-passing technique is very useful for formalizing natural language (discourse) semantics. For example Barker [2002] investigates in more details how continuized λ-terms expressing natural language meanings provide a way of dealing with phenomena related to quantification, such as scope displacement and scope ambiguity, and give a unified account of quantificational and non-quantificational noun phrases. Additionaly, Barker [2004] shows that CPS can be used to deal with focus, coordination and misplaced quantifiers.[3]

Although a continuation-passing style allows many apparently non-compositional phenomena to be handled compositionally, it requires type-raising (e.g. an atomic type α becomes $((\alpha \to o) \to o)$) and thus result in more complex and less natural λ-terms and types. However, due to the similarity between type-raising and the double-negation translation (e.g. an atomic proposition P becomes $((\mathrm{P} \to \bot) \to \bot)$), it is possible to obtain more concise and more intuitive lambda terms and types in natural languages semantics by using extended lambda calculi, such as the $\lambda\mu$-calculus [Parigot, 1992], originally developed to exploit the double negation translation and provide a Curry-Howard isomorphism for classical logic. This was done in [de Groote, 2001b], where ambiguities in the scope of natural

[2] See Section 1.4.

[3] Other interesting continuation-based analyses of different natural language phenomena include [Shan, 2002; Shan, 2004; Barker and Shan, 2008].

language quantifiers were nicely related to the non-confluence of the $\lambda\mu$-calculus (with β-, μ- and μ'-reductions), which corresponds to the non-confluence of cut-elimination in classical logic.

De Groote [2006] proposes a continuation-based approach that played a major role in the development of this dissertation, particularly for defining continuation-based dynamic logic (with exceptions), introduced in Chapters 4 and 6. This continuation-based approach offered a way to use continuations in order to implement the influential pragmatic ideas of Stalnaker [1978] in Montague's settings. This is done by providing two additional arguments to each λ-term interpreting a natural language proposition. One argument stands for the continuation of the term. Another argument represents a (previously computed) context. Moreover, within the body of the λ-term, the continuation is applied to the (possibly updated) context (therefore, the continuation is dependent on the context). This is inspired by Stalnaker's idea that each assertion is made in context, is dependent upon the context and updates the context.[4]

[4] [Heim, 1982] and [Kamp, 1981] also elaborate the work of Stalnaker [1978]. They do it, however, in a fundamentally different, non-compositional, way, as discussed in Section 2.2.

This chapter formally presented the notion of continuation and discussed the major benefits a continuation-passing technique provides to a language, such as explicit control over the evaluation of expressions. The continuation-passing technique was introduced in programming languages semantics to give compositional interpretations for full jumps, dynamic and apparently non-compositional phenomena. Therefore, this technique is promising for handling dynamic phenomena of natural language that, as discussed in Chapter 2, are apparently not compositional. The rest of this dissertation is devoted to developing such a continuation-based formal framework for natural language semantics.

Chapter 4

Continuation-based Dynamic Logic

Natural language dynamics poses, as discussed in Chapter 2, numerous challenges for formalizing natural language semantics in a truly compositional way. The difficulty lies in the fact that meaning is a function of context, i.e. of some knowledge previously obtained. The meaning of the sentence *He whistles*, for example, has to be evaluated with respect to a context to obtain the antecedent of the pronoun. Thus, if the previous sentence is *A man walks in the park*, the context, after processing the sentence, would contain an obvious referent for the pronoun *he* in the subsequent sentence *He whistles*. The approaches discussed in Section 2.2 fail to explicitly assign this context dependence to lexical items. This either undermines the compositionality of the approach (as in the case of the original DRT) or prevents the manipulation of the context (as in the case of DMG), which is necessary, for example, for handling presuppositions and conversational implicatures.

The framework that this dissertation develops not only specially focuses on the explicit representation of context, but also defines the meanings of natural languages constituents, including lexical items, in a way that they are functions of context when necessary. Another advantage of the framework is that the content of expressions is separated from the context. This makes it flexible to potentially handle pragmatic phenomena.

The framework is an extension of Montague's semantics and is truly compositional. As Montague's semantics[1], it is defined on Church's simple type theory [Church, 1940], having two atomic types: ι, the type for individuals; and o, the type for propositions. However, it has an additional type parameter γ and context is defined as a term of this type. Natural language sentences are then defined as functions of the context. Moreover, the framework uses continuation-passing techniques[2] for compositionally treating dynamic phenomena and updating the

[1] See Section 1.4.
[2] See Chapter 3.

context as the computation proceeds.

The basic idea of the framework originated in [de Groote, 2006], where a simplified framework, called here G_0, is used for handling cross-sentential and donkey anaphora. Section 4.1 is devoted to explaining de Groote's framework G_0 in detail. The meanings assigned to expressions according to this approach are, however, lengthy and not always intuitive.

This motivated de Groote to pursue more compact interpretations by idealizing a continuation-based logic. Section 4.2 develops this logic in detail. The natural language semantics framework based on this logic is called here G. It is, moreover, proven in Section 4.2 that static propositions and their dynamic counterparts obtained in G hold in the same models.

Section 4.3 presents innovations of this dissertation by developing framework GL. It elaborates de Groote's frameworks G_0 and G by putting more emphasis on context and by taking into account that context contains some knowledge (such as common knowledge and knowledge learned during the process of computation of the meaning). Due to this more realistic notion of context, it becomes possible to retrieve from it antecedents of referring expressions based on their descriptive contents. Moreover, dynamic individuals are defined as being functions of context. This is an important novelty, as it makes it possible to give direct (not type-raised) interpretations to, for example, pronouns in a way that these interpretations explicitly encode their anaphoric nature.

As in framework G, the translation from static terms to their dynamic equivalents is defined in a modular way by employing a continuation-based logical language. However, framework GL is superior to G as it additionally defines a systematic translation of non-logical constants. Finally, a more sophisticated conservation theorem for framework GL, saying that a proposition is true in a model if and only if its dynamic variant is true in the same model is also proved.

4.1 Framework G_0: Initial Approach

Philippe de Groote [2006] showed that it is possible to handle dynamic phenomena of natural language by standard tools of mathematical logic, such as simply-typed lambda calculus, and, therefore, stay within Montague's program. This is accomplished in de Groote's framework, called here G_0, by providing Montague semantics with a notion of context in a systematic and precise way.

The meaning of a sentence is a function of the context. It can be expressed in lambda calculus by defining the term standing for the interpretation of a sentence as an abstraction over a variable standing for the context. This variable is of a particular type parameter γ:

Definition 4.1. [Context, Environment] A **context** or **environment** is a term of type γ that stores the essential information from what has already been processed in the computation of the meaning of the whole discourse.

In order to make the framework flexible, the context type γ is a **parameter**, which can define any complex type. Therefore, there is no restriction on the representation of context. One can define it as a simple structure focusing on a particular phenomenon. As Montague put it:

> In interpreting a pragmatic language L we shall have to take into account the possible contexts of use. It is not necessary to consider them in their full complexity; we may instead confine our attention to those among their features which are relevant to the discourse in question. Thus it will suffice to specify the set of all complexes of relevant aspects of intended possible contexts of use. [Montague, 1968]

On the other hand, the abstract representation of context can become more sophisticated as more complex phenomena (e.g. presuppositions and implicatures) are taken into account. Since the context type γ is a parameter, the context can be easily elaborated without affecting the core of the framework.

A sentence can have a potential to change (or update) the context. The updated context has to be passed as an argument to the meaning of the subsequent sentence. In order to do so compositionally, de Groote used the notion of continuation:[3] the meaning of a sentence not only is a function of context, but also is a function of a continuation with respect to the computation of the meaning of the whole discourse. Within the body of the term standing for the meaning of a sentence, the continuation is given the possibly updated context and returns a proposition. Therefore, the continuation has type $(\gamma \to o)$.

Definition 4.2. [Continuation] A **continuation** is a term of type $(\gamma \to o)$ that denotes what is still to be processed in the computation of the meaning of the whole discourse.

Thus, a sentence is dynamically interpreted as a function that takes a context e of type γ and a continuation ϕ of type $(\gamma \to o)$ and returns a proposition:

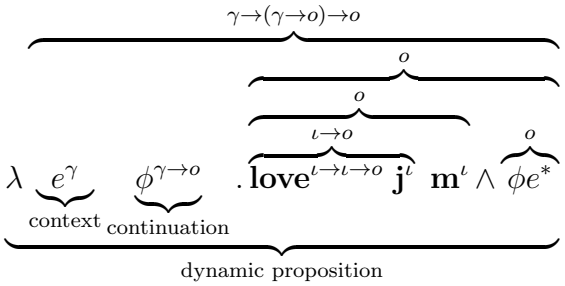
$$[\![s]\!] = \underbrace{\gamma}_{\text{context}} \to \underbrace{(\gamma \to o)}_{\text{continuation}} \to \underbrace{o}_{\text{proposition}}$$

Type $(\gamma \to (\gamma \to o) \to o)$ is, therefore, defined to be the type of a dynamic proposition.

Example 4.3. The meaning of the sentence (37) is the λ-term (4.1):

(37) *John loves Mary.*

[3] See Chapter 3.



(4.1)

where e^* is the context obtained by updating e. □

Note the presence of the conjunct ϕe^* in (4.1) that conveys that an updated context is passed as an argument to the continuation of a proposition, and is, therefore, accessible in the rest of the computation. This kind of conjunct is a subterm of every proposition in de Groote's framework.

In G_0 each object is interpreted as a variable (i.e. as a term of type ι). The framework is presented on the phenomena of cross-sentential and donkey anaphora and the type of context γ is defined as a list of individuals for the sake of simplicity:[4]

$$\gamma \doteq \texttt{list of } [\ \iota\ ] \tag{4.2}$$

Thus, the context stores only interpretations of objects that previously occurred in the discourse. When a new object is interpreted as an individual x, the current context e is updated with x, resulting in $(x :: e)$, where :: is a list constructor of type $(\iota \to \gamma \to \gamma)$ (i.e. it is a function that takes an individual and a context and returns an (updated) context).[5]

Therefore, returning to Example 4.3, e^* is $(\mathbf{m} :: \mathbf{j} :: e)$. Hence, the interpretation of Sentence (37) is as follows:

$$\lambda e\phi.\mathbf{love}\ \mathbf{j}\ \mathbf{m} \wedge \phi(\mathbf{m} :: \mathbf{j} :: e) \tag{4.3}$$

[4]The fact that γ is a parameter and, therefore, the framework is open to incorporate any desirable representation of the context is an important advantage of de Groote's approach. By defining γ as a list of individuals in [2006], de Groote consciously used a simplified notion of context in order to emphasize that the focus of his work is to formalize dynamic phenomena in a compositional framework in the spirit of Montague and is not to define a specific anaphora resolution algorithm. Importantly, de Groote pays special attention to providing a function sel responsible for anaphora resolution with necessary arguments to retrieve an antecedent in a compositional manner. How exactly sel is implemented is left open, which is, once again, an advantage of the approach: γ can be as complex as one wishes and sel can implement any anaphora resolution algorithm. The simplified notion of context de Groote [2006] used was sufficient for demonstrating how discourses containing cross-sentential and donkey anaphora can be compositionally treated. This is sometimes misunderstood and de Groote's work is wrongly claimed to have an "inadequate treatment of anaphora resolution" [Martin and Pollard, 2010]. In fact, de Groote's work, having a different objective (that of building compositional semantics), simply does not provide any treatment of anaphora resolution (but does provide the framework for its implementation).

[5]Operation :: is right associative. For example, $(x :: y :: e)$ is equivalent to $(x :: (y :: e))$.

Term (4.3) has to be computed compositionally from lexical meanings $[\![John]\!]$, $[\![Mary]\!]$ and $[\![loves]\!]$. Particularly, it has to be the result of normalizing $[\![loves]\!][\![Mary]\!][\![John]\!]$.[6]

A noun phrase in Montague semantics is a term taking a property as an argument and returning a proposition. In framework G_0 there should be two additional arguments for a term to return a proposition. Therefore, everywhere where a term of type o occurs in Montague's interpretation, there has to be a term of type $(\gamma \to (\gamma \to o) \to o)$ in de Groote's interpretation, as can be easily seen comparing (4.4a) and (4.4b), where Ω is an abbreviation for $(\gamma \to (\gamma \to o) \to o)$. Thus, a noun phrase is interpreted as a function of three arguments (a property, a context and a continuation) that returns a proposition, as can be more easily seen in (4.4c), where no abbreviation is used:

$$[\![np]\!] =_{Montague} \underbrace{(\iota \to o)}_{\substack{\text{static}\\ \text{property}}} \to \underbrace{o}_{\substack{\text{static}\\ \text{proposition}}} \tag{4.4a}$$

$$[\![np]\!] =_{de\ Groote} \underbrace{(\iota \to \Omega)}_{\substack{\text{dynamic}\\ \text{property}}} \to \underbrace{\Omega}_{\substack{\text{dynamic}\\ \text{proposition}}} \tag{4.4b}$$

$$[\![np]\!] =_{de\ Groote} \underbrace{(\iota \to \gamma \to (\gamma \to o) \to o)}_{\substack{\text{dynamic}\\ \text{property}}} \to \underbrace{\underbrace{\gamma}_{\text{context}} \to \underbrace{(\gamma \to o)}_{\text{continuation}} \to \underbrace{o}_{\text{proposition}}}_{\substack{\text{dynamic}\\ \text{proposition}}} \tag{4.4c}$$

The interpretation of *Mary*, for example, is as follows:

$$[\![Mary]\!] = \lambda \underbrace{\mathbf{P}^{\iota \to \gamma \to (\gamma \to o) \to o}}_{\substack{\text{dynamic}\\ \text{property}}} . \underbrace{\lambda \underbrace{e^{\gamma}}_{\text{context}} \underbrace{\phi^{\gamma \to o}}_{\text{continuation}} . \overbrace{\overbrace{\mathbf{P}\mathbf{m}^{\iota}}^{\gamma \to (\gamma \to o) \to o}\ e\ \overbrace{(\lambda e'^{\gamma} . \overbrace{\phi(\overbrace{\mathbf{m} :: e'}^{\gamma})}^{o})}^{\gamma \to o}}^{(\gamma \to o) \to o}}_{\substack{\text{dynamic}\\ \text{proposition}}} \tag{4.5}$$

The interpretation of *John* is analogous:

$$[\![John]\!] = \lambda \mathbf{P} . \lambda e \phi . \mathbf{P}\mathbf{j} e (\lambda e' . \phi(\mathbf{j} :: e')) \tag{4.6}$$

[6] Recall syntactic tree on Figure 1.6.

A transitive verb is interpreted in Montague semantics as a term taking two type-raised individuals and returning a proposition. Since in de Groote's framework there has to be an abstraction over a context and a continuation to get a proposition, everywhere where a term of type o occurs in Montague's interpretation, there has to be a term of type $(\gamma \to (\gamma \to o) \to o)$ in de Groote's interpretation. This can be seen comparing types in (4.7):

$$[\![tv]\!] =_{Montague} (\underbrace{(\iota \to o)}_{\text{property}} \to \underbrace{o}_{\text{proposition}}) \to (\underbrace{(\iota \to o)}_{\text{property}} \to \underbrace{o}_{\text{proposition}}) \to \underbrace{o}_{\text{proposition}} \tag{4.7a}$$

$$[\![tv]\!] =_{de\ Groote} (\underbrace{(\iota \to \Omega)}_{\text{property}} \to \underbrace{\Omega}_{\text{proposition}}) \to (\underbrace{(\iota \to \Omega)}_{\text{property}} \to \underbrace{\Omega}_{\text{proposition}}) \to \underbrace{\Omega}_{\text{proposition}} \tag{4.7b}$$

Then the interpretation of *loves* is as follows:

$$[\![loves]\!] = \lambda \mathbf{Y}^{(\iota\to\Omega)\to\Omega} \mathbf{X}^{(\iota\to\Omega)\to\Omega}.\, \mathbf{X}(\lambda x.\, \mathbf{Y}(\lambda y.(\lambda e'^{\gamma} \phi^{\gamma\to o}.\, \overbrace{\mathbf{love}^{\iota\to\iota\to o} x^{\iota}\, y^{\iota}}^{\iota\to o} \wedge \overbrace{\phi e'}^{o}))) \tag{4.8}$$

Example 4.4. [**D**, *John loves Mary*] Now, given lexical interpretations (4.8), (4.5) and (4.6) of *loves*, *Mary* and *John* respectively, the meaning (4.3) of Sentence (37) can be computed compositionally according to the parse tree in Figure 1.6, for convenience repeated in Figure 4.1:

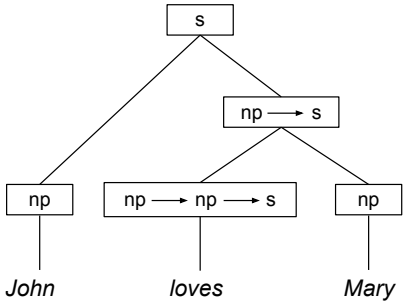


Figure 4.1: Syntactic parse tree of sentence *John loves Mary*.

$$
\begin{aligned}
\mathbf{D} &= [\![loves]\!][\![Mary]\!][\![John]\!] \\
&= (\lambda \mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.(\lambda e'\phi.\mathbf{love}xy \wedge \phi e')))) [\![Mary]\!][\![John]\!] \\
&\rightarrow_\beta (\lambda \mathbf{X}.\mathbf{X}(\lambda x.[\![Mary]\!](\lambda y.(\lambda e'\phi.\mathbf{love}xy \wedge \phi e')))) [\![John]\!] \\
&\rightarrow_\beta [\![John]\!](\lambda x.[\![Mary]\!](\lambda y.(\lambda e'\phi.\mathbf{love}xy \wedge \phi e'))) \\
&= [\![John]\!](\lambda x.(\lambda \mathbf{P}.\lambda e\phi.\mathbf{Pm}e(\lambda e.\phi(\mathbf{m} :: e')))(\lambda y.(\lambda e'\phi.\mathbf{love}xy \wedge \phi e'))) \\
&\rightarrow_\beta [\![John]\!](\lambda x.\lambda e\phi.(\lambda y.(\lambda e'\phi.\mathbf{love}xy \wedge \phi e'))\mathbf{m}e(\lambda e'.\phi(\mathbf{m} :: e'))) \\
&\rightarrow_\beta [\![John]\!](\lambda x.\lambda e\phi.(\lambda e'\phi.\mathbf{love}x\mathbf{m} \wedge \phi e')e(\lambda e'.\phi(\mathbf{m} :: e'))) \\
&\rightarrow^*_\beta [\![John]\!](\lambda x.\lambda e\phi.\mathbf{love}x\mathbf{m} \wedge (\lambda e'.\phi(\mathbf{m} :: e')e)) \\
&\rightarrow_\beta [\![John]\!](\lambda x.\lambda e\phi.\mathbf{love}x\mathbf{m} \wedge \phi(\mathbf{m} :: e)) \\
&= (\lambda \mathbf{P}.\lambda e\phi.\mathbf{Pj}e(\lambda e'.\phi(\mathbf{j} :: e')))(\lambda x.\lambda e\phi.\mathbf{love}x\mathbf{m} \wedge \phi(\mathbf{m} :: e)) \\
&\rightarrow_\beta \lambda e\phi.(\lambda x.\lambda e\phi.\mathbf{love}x\mathbf{m} \wedge \phi(\mathbf{m} :: e))\mathbf{j}e(\lambda e'.\phi(\mathbf{j} :: e')) \\
&\rightarrow^*_\beta \lambda e\phi.\mathbf{lovejm} \wedge (\lambda e'.\phi(\mathbf{j} :: e'))(\mathbf{m} :: e) \\
&\rightarrow_\beta \lambda e\phi.\mathbf{lovejm} \wedge \phi(\mathbf{j} :: \mathbf{m} :: e)
\end{aligned} \tag{4.9}
$$

□

To cope with anaphora, the context has to be accessed. This is accomplished by a special function sel of type $(\gamma \rightarrow \iota)$ that takes a context and returns an individual. The function sel is assumed to implement an anaphora resolution algorithm and to work as an oracle always retrieving the correct antecedent. This allows to interpret pronouns as shown, for example, for *he* below:

$$
[\![he]\!] = \lambda \mathbf{P}^{\overbrace{\iota \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o}^{(\iota \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o) \rightarrow \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o}}.\lambda e^{\gamma}\phi^{\overbrace{\gamma \rightarrow o}^{(\gamma \rightarrow o) \rightarrow o}}.\ \mathbf{P}(\overbrace{\mathsf{sel}_{he}e}^{\iota})\ e\ \phi \tag{4.10}
$$

Example 4.5. [**S**, *He smiles at her*] The meaning of the sentence (38) computed in accordance with the parse-tree shown in Figure 4.2 is as follows:

(38) *He smiles at her.*

$$
\begin{aligned}
\mathbf{S} &= [\![smiles_at]\!][\![her]\!][\![he]\!] \\
&= (\lambda \mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.(\lambda e'\phi.\mathbf{smile}xy \wedge \phi e')))) [\![her]\!][\![he]\!] \\
&\rightarrow_\beta (\lambda \mathbf{X}.\mathbf{X}(\lambda x.[\![her]\!](\lambda y.(\lambda e'\phi.\mathbf{smile}xy \wedge \phi e')))) [\![he]\!] \\
&\rightarrow_\beta [\![he]\!](\lambda x.[\![her]\!](\lambda y.(\lambda e'\phi.\mathbf{smile}xy \wedge \phi e'))) \\
&= [\![he]\!](\lambda x.(\lambda \mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}_{her}e)e\phi)(\lambda y.(\lambda e'\phi.\mathbf{smile}xy \wedge \phi e'))) \\
&\rightarrow_\beta [\![he]\!](\lambda x.(\lambda e\phi.(\lambda y.(\lambda e'\phi.\mathbf{smile}xy \wedge \phi e'))(\mathsf{sel}_{her}e)e\phi)) \\
&\rightarrow_\beta [\![he]\!](\lambda x.(\lambda e\phi.(\lambda e'\phi.\mathbf{smile}x(\mathsf{sel}_{her}e) \wedge \phi e')e\phi))
\end{aligned}
$$

$$
\begin{aligned}
&\to^*_\beta [\![he]\!](\lambda x.(\lambda e\phi.\mathbf{smile}x(\mathsf{sel}_{her}e) \wedge \phi e)) \\
&\quad = (\lambda \mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}_{he}e)e\phi)(\lambda x.(\lambda e\phi.\mathbf{smile}x(\mathsf{sel}_{her}e) \wedge \phi e)) \\
&\to_\beta \lambda e\phi.(\lambda x.(\lambda e\phi.\mathbf{smile}x(\mathsf{sel}_{her}e) \wedge \phi e))(\mathsf{sel}_{he}e)e\phi \\
&\to_\beta \lambda e\phi.(\lambda e\phi.\mathbf{smile}(\mathsf{sel}_{he}e)(\mathsf{sel}_{her}e) \wedge \phi e)e\phi \\
&\to^*_\beta \lambda e\phi.\mathbf{smile}(\mathsf{sel}_{he}e)(\mathsf{sel}_{her}e) \wedge \phi e \qquad (4.11)
\end{aligned}
$$

□

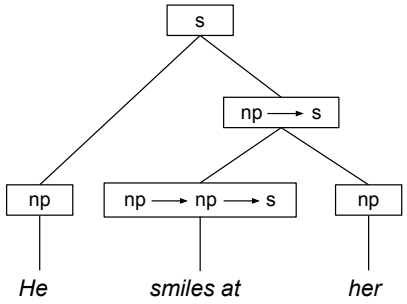


Figure 4.2: Syntactic parse tree of sentence *He smiles at her*.

As Example 4.5 shows, Sentence (38) is meaningful in de Groote's approach in the sense that it has an interpretation (4.11). However, the function sel can return individuals for *he* and *her* only when the sentence is evaluated over some context containing the corresponding antecedents. This can be done when the sentence is uttered in a discourse and the representation of this discourse is already computed. When the meaning of the discourse is updated with the meaning of the sentence, the pronominal anaphora is resolved.

Discourses in [de Groote, 2006] are, like sentences, interpreted as terms of type $(\gamma \to (\gamma \to o) \to o)$. The update of a discourse interpreted as $\mathbf{D}$ with a sentence interpreted as $\mathbf{S}$ results in interpretation $\mathsf{upd}\ \mathbf{D}\ \mathbf{S}$ of a new discourse. This interpretation is defined by the following equation:

$$
\mathsf{upd}\ \mathbf{D}\ \mathbf{S} \doteq \lambda e^{\gamma}\phi^{\gamma\to o}.\overbrace{\mathbf{D}^{\gamma\to(\gamma\to o)\to o}}^{\gamma\to(\gamma\to o)\to o}\overbrace{e}^{o}(\lambda e'^{\gamma}.\overbrace{\mathbf{S}^{\gamma\to(\gamma\to o)\to o}}^{\gamma\to o}\overbrace{e'}^{o}\phi) \qquad (4.12)
$$

Example 4.6. [upd **D** **S**] Now interpretations (4.9) and (4.11) can be composed through equation (4.12), regarding (37) as the discourse updated with the sentence (38). This leads to the interpretation of the piece of discourse (39):

(39) *John loves Mary. He smiles at her.*

$$\begin{aligned}
&\lambda e\phi.(\overbrace{\lambda e\phi.\mathbf{lovejm} \wedge \phi(\mathbf{j} :: \mathbf{m} :: e)}^{\mathbf{D}})e(\lambda e'.(\overbrace{\lambda e\phi.\mathbf{smile}(\mathsf{sel}_{he}e)(\mathsf{sel}_{her}e) \wedge \phi e}^{\mathbf{S}})e'\phi)\\
\rightarrow^*_\beta\ &\lambda e\phi.(\lambda e\phi.\mathbf{lovejm} \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))e(\lambda e'.\mathbf{smile}(\mathsf{sel}_{he}e')(\mathsf{sel}_{her}e') \wedge \phi e')\\
\rightarrow^*_\beta\ &\lambda e\phi.\mathbf{lovejm} \wedge (\lambda e'.\mathbf{smile}(\mathsf{sel}_{he}e')(\mathsf{sel}_{her}e') \wedge \phi e')(\mathbf{j} :: \mathbf{m} :: e)\\
\rightarrow_\beta\ &\lambda e\phi.\mathbf{lovejm} \wedge \mathbf{smile}(\mathsf{sel}_{he}(\mathbf{j} :: \mathbf{m} :: e))(\mathsf{sel}_{her}(\mathbf{j} :: \mathbf{m} :: e)) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e)
\end{aligned} \tag{4.13}$$

□

Interpretation (4.13) of the discourse consisting of the utterance of (39) is computed in a compositional manner. Note that the context of the interpretation of the first sentence is passed to sel operators of the interpretation of the second sentence. Assuming that an anaphora resolution mechanism is implemented in sel, the following semantic representation of (39) is obtained:

$$\lambda e\phi.\mathbf{love\ j\ m} \wedge \mathbf{smile\ j\ m} \wedge \phi(\mathbf{j} :: \mathbf{m} :: e) \tag{4.14}$$

The context $(\mathbf{j} :: \mathbf{m} :: e)$ in (4.13) (and hence in (4.14)) is accessible for future computation. This means that the individuals **j** and **m** can serve as ancestors for anaphoric pronouns in the following sentences. However, this is not always the case. For example, assuming accessibility constraint requirements of DRT, the individuals introduced by quantifiers in Sentence (40) should not be accessible for anaphoric triggers outside of the sentence. However, they clearly should be accessible for anaphoric pronouns within the sentence.

(40) *Every farmer who owns a donkey beats it.*

This accessibility constraint can also be implemented in de Groote's approach. For example, lexical items of (40) can be assigned meanings shown in Table 4.1 that lead to the desirable interpretation of the sentence, as demonstrated below. Since the lexical interpretations are dynamic, the resulting dynamic meaning of the donkey sentence does not suffer the drawbacks of the static meaning discussed in Subsection 2.1.3.

Example 4.7. [*Every farmer who owns a donkey beats it*] The meaning of the noun phrase *a donkey* is computed by reducing the term $[\![a]\!][\![donkey]\!]$:

$$\begin{aligned}
[\![a]\!][\![donkey]\!] &= (\lambda\mathbf{PQ}.\lambda e\phi.\exists(\lambda y.\mathbf{P}ye(\lambda e'.\mathbf{Q}y(y :: e')\phi)))[\![donkey]\!]\\
&\rightarrow_\beta \lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda y.[\![donkey]\!]ye(\lambda e'.\mathbf{Q}y(y :: e')\phi))\\
&= \lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda y.(\lambda xe\phi.\mathbf{d}x \wedge \phi e)ye(\lambda e'.\mathbf{Q}y(y :: e')\phi))\\
&\rightarrow_\beta \lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda y.(\lambda e\phi.\mathbf{d}y \wedge \phi e)e(\lambda e'.\mathbf{Q}y(y :: e')\phi))\\
&\rightarrow^*_\beta \lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge (\lambda e'.\mathbf{Q}y(y :: e')\phi)e)\\
&\rightarrow_\beta \lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{Q}y(y :: e)\phi)
\end{aligned} \tag{4.15}$$

Lexical item	Syntactic category	Continuation-based interpretation in G_0
farmer	n	$\lambda xe\phi.\mathbf{f}x \wedge \phi e$
donkey	n	$\lambda xe\phi.\mathbf{d}x \wedge \phi e$
owns	np $\to$ np $\to$ s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.(\lambda e'\phi.\mathbf{o}xy \wedge \phi e')))$
beats	np $\to$ np $\to$ s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.(\lambda e'\phi.\mathbf{b}xy \wedge \phi e')))$
a	n $\to$ np	$\lambda\mathbf{PQ}.\lambda e\phi.\exists(\lambda x.\mathbf{P}xe(\lambda e'.\mathbf{Q}x(x :: e')\phi))$
every	n $\to$ np	$\lambda\mathbf{PQ}.\lambda e\phi.(\forall x.\neg(\mathbf{P}xe(\lambda e'.\neg(\mathbf{Q}x(x :: e')(\lambda e'''.\top))))) \wedge \phi e$
who	(np $\to$ s) $\to$ n $\to$ n	$\lambda\mathbf{RQ}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.\mathbf{R}(\lambda\mathbf{P}.\mathbf{P}x)e'\phi)$
it	np	$\lambda\mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}_{it}e)e\phi$

Table 4.1: Continuation-based interpretations of lexical items of the sentence *Every farmer who owns a donkey beats it* in framework G_0.

Note that in Equation (4.15) the environment passed as an argument to $\mathbf{Q}$ contains the variable y introduced by the existential quantifier. This means that this variable is available to the formula $\mathbf{Q}$. Note also that the continuation ϕ of the term (4.15) is within the scope of the existential quantifier.

The meaning of the verb phrase *owns a donkey* is computed by β-reducing the term $[\![owns]\!]([\![a]\!][\![donkey]\!])$:

$$\begin{aligned}
&[\![owns]\!]([\![a]\!][\![donkey]\!])\\
&= (\lambda\mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.(\lambda e'\phi.\mathbf{o}xy \wedge \phi e'))))([\![a]\!][\![donkey]\!])\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.([\![a]\!][\![donkey]\!])(\lambda\mathbf{y}.(\lambda e'\phi.\mathbf{o}xy \wedge \phi e')))\\
&= \lambda\mathbf{X}.\mathbf{X}(\lambda x.(\lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{Q}y(y :: e)\phi))(\lambda y.(\lambda e'\phi.\mathbf{o}xy \wedge \phi e')))\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.(\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge (\lambda y.(\lambda e'\phi.\mathbf{o}xy \wedge \phi e'))y(y :: e)\phi)))\\
&\to^*_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.(\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e))))
\end{aligned}$$

The dynamic meaning of the relative clause *who owns a donkey* is computed as follows:

$$\begin{aligned}
&[\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))\\
&= (\lambda\mathbf{RQ}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.\mathbf{R}(\lambda\mathbf{P}.\mathbf{P}x)e'\phi))([\![owns]\!]([\![a]\!][\![donkey]\!]))\\
&\to_\beta \lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.([\![owns]\!]([\![a]\!]\ [\![donkey]\!]))(\lambda\mathbf{P}.\mathbf{P}x)e'\phi)\\
&= \lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.(\lambda\mathbf{X}.\mathbf{X}(\lambda x.(\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e)))))(\lambda\mathbf{P}.\mathbf{P}x)e'\phi)\\
&\to_\beta \lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.(\lambda\mathbf{P}.\mathbf{P}x)(\lambda x.(\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e))))e'\phi)\\
&\to_\beta \lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.(\lambda x.(\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e))))xe'\phi)\\
&\to_\beta \lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.(\lambda e\phi.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e)))e'\phi)\\
&\to^*_\beta \lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e')))
\end{aligned} \tag{4.16}$$

The meaning of *farmer who owns a donkey* is computed by applying the λ-

term (4.16) to the lexical interpretation of *farmer*:

$$\begin{aligned}
&(\llbracket who\rrbracket(\llbracket owns\rrbracket(\llbracket a\rrbracket\llbracket donkey\rrbracket)))\llbracket farmer\rrbracket\\
&= (\lambda\mathbf{Q}x.\lambda e\phi.\mathbf{Q}xe(\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e'))))\llbracket farmer\rrbracket\\
&\to_\beta \lambda x.\lambda e\phi.\llbracket farmer\rrbracket xe(\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e')))\\
&= \lambda x.\lambda e\phi.(\lambda xe\phi.\mathbf{f}x \wedge \phi e)xe(\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e')))\\
&\to_\beta \lambda x.\lambda e\phi.(\lambda e\phi.\mathbf{f}x \wedge \phi e)e(\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e')))\\
&\to_\beta^* \lambda x.\lambda e\phi.\mathbf{f}x \wedge (\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e')))e\\
&\to_\beta \lambda x.\lambda e\phi.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e))
\end{aligned}$$

The dynamic meaning of the noun phrase *every farmer who owns a donkey* is computed by applying the meaning of *every* to the meaning of *farmer who owns a donkey*.

$$\begin{aligned}
&\llbracket every\rrbracket((\llbracket who\rrbracket(\llbracket owns\rrbracket(\llbracket a\rrbracket\llbracket donkey\rrbracket)))\llbracket farmer\rrbracket)\\
&= (\lambda\mathbf{PQ}.\lambda e\phi.(\forall x.\neg(\mathbf{P}xe(\lambda e'.\neg(\mathbf{Q}x(x :: e')(\lambda e'''.\top))))) \wedge \phi e)\\
&\quad((\llbracket who\rrbracket(\llbracket owns\rrbracket(\llbracket a\rrbracket\ \llbracket donkey\rrbracket)))\llbracket farmer\rrbracket)\\
&\to_\beta \lambda\mathbf{Q}.\lambda e\phi.(\forall x.\neg(((\llbracket who\rrbracket(\llbracket owns\rrbracket(\llbracket a\rrbracket\ \llbracket donkey\rrbracket)))\llbracket farmer\rrbracket)\\
&\qquad xe(\lambda e'.\neg(\mathbf{Q}x(x :: e')(\lambda e'''.\top))))) \wedge \phi e\\
&= \lambda\mathbf{Q}.\lambda e\phi.(\forall x.\neg((\lambda x.\lambda e\phi.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e)))\\
&\qquad xe(\lambda e'.\neg(\mathbf{Q}x(x :: e')(\lambda e'''.\top))))) \wedge \phi e\\
&\to_\beta \lambda\mathbf{Q}.\lambda e\phi.(\forall x.\neg((\lambda e\phi.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \phi(y :: e)))\\
&\qquad e(\lambda e'.\neg(\mathbf{Q}x(x :: e')(\lambda e'''.\top))))) \wedge \phi e\\
&\to_\beta^* \lambda\mathbf{Q}.\lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge\\
&\qquad (\lambda e'.\neg(\mathbf{Q}x(x :: e')(\lambda e'''.\top)))(y :: e)))) \wedge \phi e\\
&\to_\beta \lambda\mathbf{Q}.\lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \neg(\mathbf{Q}x(x :: y :: e)(\lambda e'''.\top))))) \wedge \phi e
\end{aligned} \tag{4.17}$$

Note that in Equation (4.17) the environment containing all the individuals with their properties collected during the computation is locally passed to the formula $\mathbf{Q}$. The continuation ϕ receives only the environment e that is passed to the term as an argument; therefore, all individuals collected during the computation of the meaning of *every farmer who owns a donkey* are not available outside the logical formula interpreting this phrase.

The meaning of the verb phrase *beats it* is computed as follows:

$$\begin{aligned}
\llbracket beats\rrbracket\llbracket it\rrbracket &= (\lambda\mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.(\lambda e'\phi.\mathbf{b}xy \wedge \phi e'))))\llbracket it\rrbracket\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.\llbracket it\rrbracket(\lambda y.(\lambda e'\phi.\mathbf{b}xy \wedge \phi e')))\\
&= \lambda\mathbf{X}.\mathbf{X}(\lambda x.(\lambda\mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}_{it}e)e\phi)(\lambda y.(\lambda e'\phi.\mathbf{b}xy \wedge \phi e')))\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.\lambda e\phi.(\lambda y.(\lambda e'\phi.\mathbf{b}xy \wedge \phi e'))(\mathsf{sel}_{it}e)e\phi)
\end{aligned} \tag{4.18}$$

$$\begin{aligned}
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.\lambda e\phi.(\lambda e'\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e')e\phi)\\
&\to^*_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda x.\lambda e\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e) \qquad (4.19)
\end{aligned}$$

Finally, the dynamic meaning of the sentence is computed by applying the term (4.19) to the term (4.17):

$$\begin{aligned}
&[\![beats]\!][\![it]\!]([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]))\\
&= (\lambda\mathbf{X}.\mathbf{X}(\lambda x.\lambda e\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e))\\
&\quad ([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]))\\
&\to_\beta ([\![every]\!](([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!]))(\lambda x.\lambda e\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e)\\
&= (\lambda\mathbf{Q}.\lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \neg(\mathbf{Q}x(x :: y :: e)(\lambda e'''.\top))))) \wedge \phi e)\\
&\quad (\lambda x.\lambda e\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e)\\
&\to_\beta \lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge\\
&\qquad \neg((\lambda x.\lambda e\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e)x(x :: y :: e)(\lambda e'''.\top))))) \wedge \phi e\\
&\to_\beta \lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge\\
&\qquad \neg((\lambda e\phi.\mathbf{b}x(\mathsf{sel}_{it}e) \wedge \phi e)(x :: y :: e)(\lambda e'''.\top))))) \wedge \phi e\\
&\to^*_\beta \lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge\\
&\qquad \neg(\mathbf{b}x(\mathsf{sel}_{it}(x :: y :: e)) \wedge (\lambda e'''.\top)(x :: y :: e))))) \wedge \phi e\\
&\to_\beta \lambda e\phi.(\forall x.\neg(\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge \mathbf{o}xy \wedge \neg(\mathbf{b}x(\mathsf{sel}_{it}(x :: y :: e)) \wedge \top)))) \wedge \phi e \qquad (4.20)
\end{aligned}$$

The resulting dynamic interpretation (4.20) of the donkey sentence is logically equivalent to (4.21):

$$\lambda e\phi.\forall(\lambda x.\mathbf{f}x \to \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \to \mathbf{b}x(\mathsf{sel}_{it}(x :: y :: e))))) \wedge \phi e \qquad (4.21)$$

Note that, in accordance with DRT's accessibility constraint, the individuals bound by quantifiers are not accessible outside the sentence. □

The dynamic formula (4.21), computed using the dynamic lexical interpretations, does not have the drawbacks of the static formula (2.15), computed using the static lexical interpretations.

First of all, the second argument of **b**, standing for the anaphoric pronoun, is not a free dummy variable, but a term $(\mathsf{sel}_{it}(y :: x :: e))$. This term consists of the selection function sel that takes as argument a context containing the available individuals "collected" during the computation. Thus, in contrast to the static case, the second argument of **b** is self-sufficient: the function sel, which implements an anaphora resolution algorithm, selects a required individual from the context. In the current case, the selection function returns the individual y, leading to the final dynamic meaning (4.22) of Sentence (40):

$$\lambda e\phi.\forall(\lambda x.\mathbf{f}x \to \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \to \mathbf{b}xy)) \wedge \phi e \qquad (4.22)$$

Moreover, in the dynamic interpretation, unlike in the static one, the formula $\mathbf{b}xy$ is within the scope of the quantifier binding the variable y, exactly as desired. Furthermore, the quantifier binding y has been changed during the computation from existential to universal, which is also impossible in the static approach. These improvements are the consequences of employing a continuation-passing technique.

The list below summarizes the advantages that de Groote's approach brought to compositional semantics:

- The approach is independent of the intermediate language[7] used to express meanings of the expressions. This allows to use mathematical and logical theories developed outside computational linguistics.[8] Therefore, natural language phenomena can be explained in terms of well-established and well-understood theories.[9]
- Variables do not have any special status and are variables in the usual mathematical sense. Therefore, the notions of free and bound variables are standard.
- There is no imperative dynamic notions as assignment functions, therefore destructive assignment problem does not hold. Meanings assigned to expressions are closed λ-terms.
- There is no need for rules that artificially extend the scope of quantifiers.
- Context and content are regarded separately, but they do interact during the computation of the meaning of discourse.
- The approach does not depend on any specific structure given to the context. In contrast, context is defined as a term of type parameter γ and, therefore, its structure can be altered when necessary.

[7] See Section 1.4.

[8] An extension of first logic language with λ is used here because it is convenient and intuitive.

[9] For a framework to be simple, clear and implementable, it is important that it is defined with compatible theories. Frameworks that ignore this principle are normally ad-hoc and, therefore, are not easily generalizable. An example is a recent work of Martin and Pollard [Martin and Pollard, 2010; Martin and Pollard, 2011]. Although their work is heavily inspired by [de Groote, 2006], they drastically deviate from de Groote's goal of having a systematic and uniform framework of natural language semantics. Focusing on enriching the notion of context, Martin and Pollard define a framework that tries to use lambda expressions (syntax) and models (semantics) at the same level. Consider, for example, their [Martin and Pollard, 2011] treatment of a pronoun *it*:

$$\lambda Ds.D(\bigsqcup_{(\mathbf{r}s)} \lambda(i : w_n)((\mathbf{c}s) \text{ entails } (\mathsf{nonhuman}(\mathbf{a}si))))s$$

In this interpretation there is a (syntactic) lambda abstraction and a (semantic) entailment relation. Their account of presupposition [Martin and Pollard, 2011] suffers the same problem.

- The approach is truly compositional: the meaning of a complex expression is computed by β-reducing the composition of the meanings of its lexical items.

4.2 Framework G

Although compositional dynamic framework G_0, introduced in [de Groote, 2006] and reviewed in the previous section, have shown itself to be promising by successfully handling donkey anaphora, its interpretations look complex and the computation of the meaning can be difficult to understand. In his later talks, de Groote proposed an improvement of framework G_0, called here framework G, that represents his semantics in a more concise and systematic way. To do so, de Groote defined a continuation-based dynamic logic and this section presents this logic.

4.2.1 Formal Details

Terms and types are given by Definitions 4.8 and 4.10:

Definition 4.8. [λ-terms] The set of **λ-terms** Λ is constructed from an enumerable set of variables $V = \{v, v_1, v_2, \dots\}$, logical constants $\wedge, \exists, \neg$, two special constants :: and sel, an enumerable set of predicate symbols $R = \{R_1, R_2, \dots\}$ and an enumerable set of constants $K = \{c, c_1, c_2, \dots\}$ using application and (function) abstraction:

$$\begin{array}{rcl} x \in V & \Longrightarrow & x \in \Lambda \\ c \in K & \Longrightarrow & c \in \Lambda \\ M, N \in \Lambda & \Longrightarrow & (MN) \in \Lambda \\ x \in V, M \in \Lambda & \Longrightarrow & (\lambda x.M) \in \Lambda \\ M \in \Lambda & \Longrightarrow & (\exists M) \in \Lambda \\ M, N \in \Lambda & \Longrightarrow & (M \wedge N) \in \Lambda \\ M \in \Lambda & \Longrightarrow & (\neg M) \in \Lambda \\ M, x \in \Lambda & \Longrightarrow & (M :: x) \in \Lambda \\ M \in \Lambda & \Longrightarrow & (\mathsf{sel}(M)) \in \Lambda \end{array}$$

Definition 4.9. [Free variables] The set of **free variables** of t, $FV(t)$, is defined inductively as follows:

$$\begin{aligned} FV(x) &= \{x\} \\ FV(c) &= \{\} \\ FV(MN) &= FV(M) \cup FV(N) \\ FV(\lambda x.M) &= FV(M) - \{x\} \end{aligned}$$

DEFINITION 4.10. [Types] The set T of types is defined inductively as follows:

$$\begin{array}{rll} \text{Atomic types:} & \iota \in \mathrm{T} & \text{(static individuals)} \\ & o \in \mathrm{T} & \text{(static propositions)} \\ & \gamma \in \mathrm{T} & \text{(context)} \\ \\ \text{Complex types:} & \alpha, \beta \in \mathrm{T} \Longrightarrow & (\alpha \to \beta) \in \mathrm{T} \end{array}$$

Typing rules define typing relations between terms and types:

DEFINITION 4.11. [Typing rules] A statement $t : \delta$, meaning t has type δ, is **derivable** from the basis Δ, i.e. $\Delta \vdash t : \delta$, if $\Delta \vdash t : \delta$ can be produced using the following rules:

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \top : o}\ \text{axiom}$$

$$\frac{}{\Gamma, M : o, N : o \vdash M \wedge N : o}$$

$$\frac{}{\Gamma, M : \iota \to o \vdash \exists M : o}$$

$$\frac{}{\Gamma, M : o \vdash \neg M : o}$$

$$\frac{}{\Gamma, c : \gamma \vdash \mathsf{sel}\ c : \iota}$$

$$\frac{}{\Gamma, i : \iota, c : \gamma \vdash (i :: c) : \gamma}$$

$$\frac{}{\Gamma \vdash c_{iv} : \iota \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{tv} : \iota \to \iota \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{n} : \iota \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{np} : (\iota \to o) \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{rp} : (((\iota \to o) \to o) \to o) \to (\iota \to o) \to (\iota \to o)}\ \text{axiom}$$

$$\frac{\Gamma \vdash v : \alpha \to \beta \qquad \Gamma \vdash u : \alpha}{\Gamma \vdash vu : \beta}\ \text{app}$$

$$\frac{\Gamma, x : \alpha \vdash v : \beta}{\Gamma \vdash \lambda x.v : \alpha \to \beta}\ \text{abs}$$

where c_{tv}, c_{iv}, c_n, c_{np} and c_{rp} are constants standing for transitive and intransitive verbs, nouns, noun phrases and relative pronouns respectively. Typing rules for other syntactic categories can be added analogously.

The first axiom and the two rules (application and abstraction) are standard typing relations in simply-typed lambda calculus. The second axiom determines the type of the logical $\top$ symbol. The constant sel has type $(\gamma \to \iota)$ and the constant :: has type $(\iota \to \gamma \to \gamma)$.

Each static type can be dynamized in the following way:

Definition 4.12. [Dynamization of types] Let ι and o be atomic types, γ be a type parameter, α and β be arbitrary types. Then the types are **dynamized** in the following way:

$$\overline{\iota} \doteq \iota \tag{4.23a}$$
$$\overline{o} \doteq \gamma \to (\gamma \to o) \to o \tag{4.23b}$$
$$\overline{\alpha \to \beta} \doteq \overline{\alpha} \to \overline{\beta} \tag{4.23c}$$

Note that the type o of a static proposition is transformed to type $(\gamma \to (\gamma \to o) \to o)$. Type $(\gamma \to (\gamma \to o) \to o)$ is thus the type of a dynamic proposition. Therefore, dynamic propositions are functions from γ (type of context) and $(\gamma \to o)$ (type of continuation) to o (type of static proposition). Static and dynamic individuals are defined to have the same type.

For each logical constant ($\neg$, $\wedge$ and $\exists$), its dynamic counterpart is specified by the following definition:

Definition 4.13. [Dynamic logical constants] Let $\mathbf{A}$ and $\mathbf{B}$ be terms of type $(\gamma \to (\gamma \to o) \to o)$, $\mathbf{P}$ be the term of type $(\iota \to \gamma \to (\gamma \to o) \to o)$, e and e' be terms of type γ, ϕ be a term of type $(\gamma \to o)$, x be the term of type ι. **Dynamic negation, conjunction and existential quantification** are defined respectively as follows:

$$\sim \mathbf{A} \doteq \lambda e\phi.\neg(\mathbf{A}e(\lambda e'.\top)) \wedge \phi e \tag{4.24a}$$
$$\mathbf{A} \curlywedge \mathbf{B} \doteq \lambda e\phi.\mathbf{A}e(\lambda e'.\mathbf{B}e'\phi) \tag{4.24b}$$
$$\Sigma(\lambda x.\mathbf{P}[x]) \doteq \lambda e\phi.\exists(\lambda x.\mathbf{P}[x]\ (x :: e)\ \phi) \tag{4.24c}$$

Dynamic negation of a dynamic proposition $\mathbf{A}$, $\sim \mathbf{A}$, is an abbreviation used for the term shown in (4.24a). Within the body of this term, the continuation and context of $\mathbf{A}$ are "erased" (by giving the term $(\lambda e'.\top)$ as the second argument of $\mathbf{A}$)[10], the resulting static proposition is negated, and the conjunct ϕe is added.

[10] This can be more clearly seen from Corollary 4.18.

Note that both ϕ and e are variables bound by λ. Therefore, the context of $\mathbf{A}$ is not available to the continuation of the resulting term $\sim \mathbf{A}$. Dynamic existentially quantified term $\Sigma(\lambda x.\mathbf{P}[x])$ is an abbreviation for the term shown in (4.24c). It has a λ-abstraction over variables e and ϕ and an existentially quantified variable x. Its body contains $\mathbf{P}[x]$, which is given e updated with x, $(x :: e)$, and ϕ as arguments. Note that $\sim$ and Σ are defined respectively via $\neg$ and $\exists$. Dynamic conjunction is defined as a composition of two dynamic terms. The logical conjunction of static propositions is provided by the fact that each dynamic proposition has a conjunct ϕe in its body, as defined in (4.25a) of the next definition.

Given dynamic logical connectives, all static terms can be dynamized:

Definition 4.14. [Dynamization of terms] Let P be a term of type $(\iota_1 \to \cdots \to \iota_n \to o)$, A and B be terms of type o, $t_1, \ldots, t_n$ and x be terms of type ι. Then propositions, negated propositions, conjunctions of two propositions and existentially quantified propositions are dynamized in the following way:

$$\overline{\mathrm{P}t_1 \ldots t_n} \doteq \lambda e\phi.\mathrm{P}t_1 \ldots t_n \wedge \phi e \tag{4.25a}$$

$$\overline{\neg \mathrm{A}} \doteq \sim \overline{\mathrm{A}} \tag{4.25b}$$

$$\overline{\mathrm{A} \wedge \mathrm{B}} \doteq \overline{\mathrm{A}} \curlywedge \overline{\mathrm{B}} \tag{4.25c}$$

$$\overline{\exists(\lambda x.\mathrm{P}[x])} \doteq \Sigma(\lambda x.\overline{\mathrm{P}[x]}) \tag{4.25d}$$

Equation (4.25a) defines the dynamization of a proposition $\mathrm{P}t_1 \ldots t_n$ of type o by adding a λ-abstraction with two arguments e and ϕ (of types γ and $(\gamma \to o)$ respectively) and a conjunct ϕe. Therefore, the resulting dynamic term is of type $(\gamma \to (\gamma \to o) \to o)$, the type of a dynamic proposition. Equations 4.25b, 4.25c and 4.25d extend dynamization to non-atomic formulas.

Note that Definitions 4.13 and 4.14 allow representing de Groote's [2006] (Section 4.1) dynamic terms in a compact way. While interpretations in [de Groote, 2006] explicitly show extra parameters, i.e. contexts and continuations, these new definitions make it possible to hide these parameters. Moreover, the resulting compact dynamic terms are structurally analogous to their original static counterparts and, hence, are more intuitive.

Remark 4.15. In equations below, terms on the left side of $\doteq$ abbreviate respective terms on the right side:

$$\mathbf{A} \Rightarrow \mathbf{B} \doteq \sim (\mathbf{A} \curlywedge \sim \mathbf{B}) \tag{4.26}$$

$$\Pi(\lambda x.\mathbf{P}[x]) \doteq \sim \Sigma(\lambda x. \sim \mathbf{P}[x]) \tag{4.27}$$

Proposition 4.16 proves an important β-equivalence that can be useful when computing interpretations of certain phrases containing an existentially quantified variable.

Proposition 4.16. *For all terms* A *and* B *of type* o *such that* $x \in FV(\mathrm{A})$ *and* $x \notin FV(\mathrm{B})$ *for an* x *of type* ι*, the following equivalence holds:*

$$\Sigma(\lambda x.\overline{\mathrm{A}[x]}) \curlywedge \overline{\mathrm{B}} =_\beta \Sigma(\lambda x.\overline{\mathrm{A}[x] \curlywedge \overline{\mathrm{B}}})$$

Proof.

$$\begin{aligned}
&\Sigma(\lambda x.\overline{\mathrm{A}[x]}) \curlywedge \overline{\mathrm{B}} \\
\to^*_\beta\ &(\lambda e\phi.\exists(\lambda x.\mathrm{A}[x] \wedge \phi(x :: e))) \curlywedge \overline{\mathrm{B}} && \text{(by (4.24c))} \\
=\ &\lambda e\phi.(\lambda e\phi.\exists(\lambda x.\mathrm{A}[x] \wedge \phi(x :: e)))e(\lambda e.\overline{\mathrm{B}}e\phi) && \text{(by (4.24b))} \\
\to^*_\beta\ &\lambda e\phi.(\lambda e\phi.\exists(\lambda x.\mathrm{A}[x] \wedge \phi(x :: e)))e(\lambda e.\mathrm{B} \wedge \phi e) && \text{(by (4.25a))} \\
\to^*_\beta\ &\lambda e\phi.\exists(\lambda x.\mathrm{A}[x] \wedge (\mathrm{B} \wedge \phi(x :: e))) && (4.28)
\end{aligned}$$

$$\begin{aligned}
&\Sigma(\lambda x.\overline{\mathrm{A}[x]} \curlywedge \overline{\mathrm{B}}) \\
=\ &\Sigma(\lambda x.\lambda e\phi.\overline{\mathrm{A}[x]}e(\lambda e.\overline{\mathrm{B}}e\phi)) && \text{(by (4.24b))} \\
\to^*_\beta\ &\Sigma(\lambda x.\lambda e\phi.(\lambda e\phi.\mathrm{A}[x] \wedge \phi e)e(\lambda e.\mathrm{B} \wedge \phi e)) && \text{(by (4.25a))} \\
\to^*_\beta\ &\Sigma(\lambda x.\lambda e\phi.\mathrm{A}[x] \wedge (\mathrm{B} \wedge \phi e)) && (4.29) \\
\to^*_\beta\ &\lambda e\phi.\exists(\lambda x.\mathrm{A}[x] \wedge (\mathrm{B} \wedge \phi(x :: e))) && \text{(by (4.24c))}
\end{aligned}$$

□

Proposition 4.17. *For all* h *and* v *of type* o*, and for all* u *of type* γ*, the following holds:*

$$\overline{h}u(\lambda e.v) = h \wedge v$$

where e *is a variable of type* γ*.*

Proof. The proof is by induction on the structure of the term h.

- h is a proposition of the form $\mathrm{P}t_1 \dots t_n$.

$$\begin{aligned}
\overline{\mathrm{P}t_1 \dots t_n}u(\lambda e.v) &= (\lambda e\phi.\mathrm{P}t_1 \dots t_n \wedge \phi e)u(\lambda e.v) && \text{(by (4.25a))} \\
&\to_\beta (\lambda\phi.\mathrm{P}t_1 \dots t_n \wedge \phi u)(\lambda e.v) \\
&\to_\beta \lambda\phi.\mathrm{P}t_1 \dots t_n \wedge (\lambda e.v)u \\
&\to_\beta \lambda\phi.\mathrm{P}t_1 \dots t_n \wedge v
\end{aligned}$$

- h is a negated proposition $\neg w$.

$$\begin{aligned}
\overline{\neg w}u(\lambda e.v) &= \sim \overline{w}u(\lambda e.v) && \text{(by (4.25b))} \\
&= (\lambda e\phi.\neg(\overline{w}e(\lambda e'.\top)) \wedge \phi e)u(\lambda e.v) && \text{(by (4.24a))} \\
&\to_\beta (\lambda\phi.\neg(\overline{w}u(\lambda e'.\top)) \wedge \phi u)(\lambda e.v) \\
&\to_\beta \neg(\overline{w}u(\lambda e'.\top)) \wedge (\lambda e.v)u \\
&\to_\beta \neg(\overline{w}u(\lambda e'.\top)) \wedge v \\
&= \neg(w \wedge \top) \wedge v && \text{(by I.H.)} \\
&= \neg w \wedge v && (4.30)
\end{aligned}$$

- h is a conjunction of two propositions $w \wedge z$.

$$\begin{aligned}
(\overline{w \wedge z})u(\lambda e.v) &= (\overline{w} \curlywedge \overline{z})u(\lambda e.v) && \text{(by (4.25c))}\\
&= (\lambda e\phi.\overline{w}e(\lambda e'.\overline{z}e'\phi))u(\lambda e.v) && \text{(by (4.24b))}\\
&\to_\beta (\lambda\phi.\overline{w}u(\lambda e'.\overline{z}e'\phi))(\lambda e.v)\\
&\to_\beta \overline{w}u(\lambda e'.\overline{z}e'(\lambda e.v))\\
&= \overline{w}u(\lambda e'.z \wedge v) && \text{(by I.H., } e \notin FV(v))\\
&= w \wedge (z \wedge v) && \text{(by I.H., } e' \notin FV(z \wedge v))\\
&\equiv (w \wedge z) \wedge v
\end{aligned}$$

- h is an existentially quantified formula of the form $\exists(\lambda x.\mathrm{P}[x])$.

$$\begin{aligned}
\overline{\exists(\lambda x.\mathrm{P}[x])}u(\lambda e.v) &= (\Sigma(\lambda x.\overline{\mathrm{P}[x]}))u(\lambda e.v) && \text{(by (4.25d))}\\
&= (\lambda e\phi.\exists(\lambda x.\mathbf{P}[x](x :: e)\phi))u(\lambda e.v) && \text{(by (4.24c))}\\
&\to_\beta (\lambda\phi.\exists(\lambda x.\mathbf{P}[x](x :: u)\phi))(\lambda e.v)\\
&\to_\beta \exists(\lambda x.\mathbf{P}[x](x :: u)(\lambda e.v))\\
&= \exists(\lambda x.\mathrm{P}[x] \wedge v) && \text{(by I.H.)}\\
&= \exists(\lambda x.\mathrm{P}[x]) \wedge v && (x \notin FV(v))
\end{aligned}$$

□

Corollary 4.18. *For all propositions t of type o, and for all terms u of type γ, the following folds:*

$$\bar{t}u(\lambda e.\top) \equiv t$$

where e is a variable of type γ.

Proof. Take v equal to $\top$ in Proposition 4.17. Then

$$\bar{t}u(\lambda e.\top) = t \wedge \top \equiv t$$

□

Definition 4.19. A dynamic proposition $\mathbf{t}$ is true in a model $\mathcal{M}$, denoted $\mathcal{M} \models_{dyn} \mathbf{t}$, if and only if $\mathcal{M} \models \mathbf{t}u(\lambda e.\top)$ for every u of type γ.

Theorem 4.20. [Conservation] *A proposition t is true in a model $\mathcal{M}$ if and only if its dynamic variant $\bar{t}$ is true in the same model:*

$$\mathcal{M} \models t \text{ iff } \mathcal{M} \models_{dyn} \bar{t}$$

Proof.

If $\mathcal{M} \models t$, then, by Corollary 4.18, $\mathcal{M} \models \bar{t}u(\lambda e.\top)$. Therefore, by Definition 4.19, $\mathcal{M} \models_{dyn} \bar{t}$.

If $\mathcal{M} \models_{dyn} \bar{t}$, then, by Definition 4.19, $\mathcal{M} \models \bar{t}u(\lambda e.\top)$. Therefore, by Corollary 4.18, $\mathcal{M} \models t$. □

The conservation theorem proves that a static proposition and its dynamic version defined in this section hold in the same models.

4.2.2 Donkey Sentences

Tables 4.2 and 4.3 show respectively static and dynamic (according to G) interpretations for the lexical items in the donkey sentence (41):

(41) *Every farmer who owns a donkey beats it.*

Note that the type of every dynamic term is analogous to its static type. The only difference is that each atomic type of a dynamic term is dynamized according to Definition 4.12. All terms in Table 4.3, except the interpretation of the pronoun *it*, are dynamized following the rules in Definition 4.14. These rules allow the presentation of dynamic terms in a compact way. They ensure that dynamic terms are structurally analogous to their static counterparts and, therefore, are more intuitive. The dynamic interpretation of *it* is constructed not by directly following the dynamization rules, because *it* is a unconventional lexical item: there is an anaphor to be solved. Therefore, $\widetilde{[\![it]\!]}$[11] contains the selection function sel that takes a context (from which a referent has to be chosen) as an argument.

Taking these dynamic interpretations to compute the meaning of Sentence (41), term (4.31) β-reduces to term (4.32), which normalizes to (4.33):

$$\overline{[\![beats]\!]}\ \widetilde{[\![it]\!]}(\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]})) \tag{4.31}$$

$$\rightarrow^*_\beta \Pi(\lambda x.(\bar{\mathbf{f}}x \curlywedge \Sigma(\lambda z.\bar{\mathbf{d}}z \curlywedge \bar{\mathbf{o}}xz)) \Rightarrow (\lambda e\phi.\bar{\mathbf{b}}x(\mathsf{sel}_{it}e)e\phi)) \tag{4.32}$$

$$\rightarrow^*_\beta \lambda e\phi.\forall(\lambda x.\mathbf{f}x \rightarrow \forall(\lambda z.(\mathbf{d}z \wedge \mathbf{o}xz) \rightarrow \mathbf{b}x(\mathsf{sel}_{it}(x :: z :: e)))) \wedge \phi e \tag{4.33}$$

Resulting term (4.33) is equivalent to (4.21) obtained in framework G_0 interpretations. Indeed, framework G is equivalent to de Groote's [2006] framework G_0. However, it is advantageous over G_0 due to the compact notations for dynamic terms. These notations significantly systematize the framework and make the interpretations more concise and intuitive. Moreover, the systematic translations of static terms into dynamic terms makes it possible to prove a conservation result 4.20 for G, that guarantees that static and dynamic interpretations are satisfied in the same models.

[11] Here and further on, dynamic interpretations of unconventional lexical items are marked with tilde.

Lexical item	Syntactic category	Static type	Static interpretation
farmer	n	$\iota \to o$	$\mathbf{f}$
donkey	n	$\iota \to o$	$\mathbf{d}$
owns	np $\to$ np $\to$ s	$((\iota \to o) \to o) \to ((\iota \to o) \to o) \to o$	λYX.X(λx.Y($\lambda y.\mathbf{o}xy$))
beats	np $\to$ np $\to$ s	$((\iota \to o) \to o) \to ((\iota \to o) \to o) \to o$	λYX.X(λx.Y($\lambda y.\mathbf{b}xy$))
every	n $\to$ np	$(\iota \to o) \to ((\iota \to o) \to o)$	λPQ.$\forall(\lambda x.\mathrm{P}x \to \mathrm{Q}x)$
a	n $\to$ np	$(\iota \to o) \to ((\iota \to o) \to o)$	λPQ.$\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)$
who	(np $\to$ s) $\to$ n $\to$ n	$(((\iota \to o) \to o) \to o) \to (\iota \to o) \to (\iota \to o)$	λRQx.Q$x \wedge$ R(λP.Px)
it	np	$(\iota \to o) \to o$	λP.P?

Table 4.2: Static lexical interpretations.

Lexical item	Syntactic category	Dynamic type	Dynamic interpretation in G
farmer	n	$\bar{\iota} \to \bar{o}$	$\bar{\mathbf{f}}$
donkey	n	$\bar{\iota} \to \bar{o}$	$\bar{\mathbf{d}}$
owns	np $\to$ np $\to$ s	$((\bar{\iota} \to \bar{o}) \to \bar{o}) \to ((\bar{\iota} \to \bar{o}) \to \bar{o}) \to \bar{o}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.\bar{\mathbf{o}}xy))$
beats	np $\to$ np $\to$ s	$((\bar{\iota} \to \bar{o}) \to \bar{o}) \to ((\bar{\iota} \to \bar{o}) \to \bar{o}) \to \bar{o}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda x.\mathbf{Y}(\lambda y.\bar{\mathbf{b}}xy))$
every	n $\to$ np	$(\bar{\iota} \to \bar{o}) \to ((\bar{\iota} \to \bar{o}) \to \bar{o})$	$\lambda\mathbf{PQ}.\Pi(\lambda x.\mathbf{P}x \Rightarrow \mathbf{Q}x)$
a	n $\to$ np	$(\bar{\iota} \to \bar{o}) \to ((\bar{\iota} \to \bar{o}) \to \bar{o})$	$\lambda\mathbf{PQ}.\Sigma(\lambda x.\mathbf{P}x \curlywedge \mathbf{Q}x)$
who	(np $\to$ s) $\to$ n $\to$ n	$(((\bar{\iota} \to \bar{o}) \to \bar{o}) \to \bar{o}) \to (\bar{\iota} \to \bar{o}) \to (\bar{\iota} \to \bar{o})$	$\lambda\mathbf{RQ}x.\mathbf{Q}x \curlywedge \mathbf{R}(\lambda\mathbf{P}.\mathbf{P}x)$
it	np	$(\bar{\iota} \to \bar{o}) \to \bar{o}$	$\lambda\mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}_{it}e)e\phi$

Table 4.3: Dynamic lexical interpretations in framework G.

4.3 Framework GL

Framework G introduced in Section 4.3 can still be elaborated. First of all, the prerequisites for anaphora resolution can be made more realistic by making the descriptive content of the pronoun to be an additional argument of the function sel. Moreover, in the long term it is desirable to handle more complex phenomena, as, for example, presuppositions and conversational implicatures. Not only this requires more elaborated structure of context, but also the context has to be treated in a more sophisticated way. This section develops a more powerful dynamic framework GL of compositional semantics in the spirit of the frameworks G_0 and G presented in Sections 4.1 and 4.2 respectively.

4.3.1 Formal Details

Definition 4.21. [λ-terms] The set of **λ-terms** Λ is constructed from an enumerable set of variables $V = \{v, v_1, v_2, \dots\}$, logical constants $\wedge, \exists, \neg$, two special constants upd and sel, and an enumerable set of constants $K = \{c, c_1, c_2, \dots\}$ using application and (function) abstraction:

$$\begin{array}{rcll} x \in V & \Longrightarrow & x \in \Lambda & \\ c \in K & \Longrightarrow & c \in \Lambda & \\ M, N \in \Lambda & \Longrightarrow & (MN) \in \Lambda & \text{(application)} \\ x \in V, M \in \Lambda & \Longrightarrow & (\lambda x.M) \in \Lambda & \text{(abstraction)} \end{array}$$

Remark 4.22. Symbols $\vee$, $\forall$ and $\rightarrow$ abbreviate the following terms:

$$\begin{aligned} \vee &\doteq \lambda \mathrm{AB}.\neg(\neg \mathrm{A} \wedge \neg \mathrm{B}) \\ \forall &\doteq \lambda \mathrm{P}.\neg\exists(\lambda x.(\neg \mathrm{P}x)) \\ \rightarrow &\doteq \lambda \mathrm{AB}.\neg(\mathrm{A} \wedge \neg \mathrm{B}) \end{aligned}$$

Definition 4.23. [Free variables] The set of **free variables** of t, $FV(t)$, is defined inductively as follows:

$$\begin{aligned} FV(x) &= \{x\} \\ FV(c) &= \{\} \\ FV(MN) &= FV(M) \cup FV(N) \\ FV(\lambda x.M) &= FV(M) - \{x\} \end{aligned}$$

Definition 4.24. [Types] The set T of types of the continuation-based dynamic logic is defined inductively as follows:

$$\begin{array}{rll} \text{Atomic types:} & \iota \in \mathrm{T} & \text{(static individual)} \\ & o \in \mathrm{T} & \text{(static proposition)} \\ & \gamma \in \mathrm{T} & \text{(context)} \end{array}$$

$$\text{Complex types:} \quad \alpha, \beta \in \mathrm{T} \Longrightarrow \ (\alpha \to \beta) \in \mathrm{T}$$

Definition 4.25. [Typing rules] A statement $t : \delta$ is **derivable** from the basis Δ, i.e. $\Delta \vdash t : \delta$, if $\Delta \vdash t : \delta$ can be produced using the following rules:

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \top : o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \wedge : o \to o \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \exists : (\iota \to o) \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \neg : o \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \mathsf{sel} : (\iota \to o) \to \gamma \to \iota}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash \mathsf{upd} : o \to \gamma \to \gamma}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{iv} : \iota \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{tv} : \iota \to \iota \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{n} : \iota \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{np} : (\iota \to o) \to o}\ \text{axiom}$$

$$\frac{}{\Gamma \vdash c_{rp} : (((\iota \to o) \to o) \to o) \to (\iota \to o) \to (\iota \to o)}\ \text{axiom}$$

$$\frac{\Gamma \vdash v : \alpha \to \beta \qquad \Gamma \vdash u : \alpha}{\Gamma \vdash vu : \beta}\ \text{app}$$

$$\frac{\Gamma, x : \alpha \vdash v : \beta}{\Gamma \vdash \lambda x.v : \alpha \to \beta}\ \text{abs}$$

where c_{tv}, c_{iv}, c_n, c_{np} and c_{rp} are constants standing for transitive and intransitive verbs, nouns, noun phrases and relative pronouns respectively. Typing rules for other syntactic categories can be added analogously.

Definition 4.26. [Dynamization of types] Let ι and o be atomic types, γ be a type parameter, α and β be any types. Then the types are **dynamized** in the following way:

$$\bar{\iota} \doteq \gamma \to \iota \tag{4.35a}$$
$$\bar{o} \doteq \gamma \to (\gamma \to o) \to o \tag{4.35b}$$
$$\overline{\alpha \to \beta} \doteq \bar{\alpha} \to \bar{\beta} \tag{4.35c}$$

Thus, additionally to parametrizing a proposition of type o with a context of type γ and a continuation of type $(\gamma \to o)$, as in [de Groote, 2006], Definition 4.26 parametrizes terms of type ι with a context.

Definition 4.28 below formalizes a systematic translation of every term t to its dynamic equivalent $\bar{t}$. The translation is relatively straightforward for all kinds of terms except non-logical constants. Non-logical constants have various types, therefore their dynamization should also be defined. This is accomplished with dynamization $\mathbb{D}_\delta$ and reading $\mathbb{S}_\delta$ functions defined immediately below. The dynamization function transforms static terms into equivalent dynamic terms. Reading function is a function that transforms dynamic terms (obtained via $\mathbb{D}_\delta$) into original static terms. Note that the dynamization function and the reading functions are mutually dependent.

Definition 4.27. [Dynamization and reading functions] **Dynamization function** is a function $\mathbb{D}_\delta$ that has type $((\gamma \to \delta) \to \bar{\delta})$ and is inductively defined on the type δ as follows:

$$\mathbb{D}_\iota[\mathtt{a}] \doteq \mathtt{a} \tag{4.36a}$$
$$\mathbb{D}_o[\mathrm{P}] \doteq \lambda e\phi.\mathrm{P}e \wedge \phi(\mathsf{upd}(\mathrm{P}e, e)) \tag{4.36b}$$
$$\mathbb{D}_{\alpha\to\beta}[f] \doteq \lambda\mathbf{a}.\mathbb{D}_\beta[\lambda e.f\ e\ \mathbb{S}_\alpha[\mathbf{a}, e]] \tag{4.36c}$$

Reading function is a function $\mathbb{S}_\delta$ that has type $(\bar{\delta} \to \gamma \to \delta)$ and is inductively defined on the type δ as follows:

$$\mathbb{S}_\iota[\mathbf{a}, e] \doteq \mathbf{a}e \tag{4.37a}$$
$$\mathbb{S}_o[\mathbf{P}, e] \doteq \mathbf{P}e(\lambda e.\top) \tag{4.37b}$$
$$\mathbb{S}_{\alpha\to\beta}[\mathbf{f}, e] \doteq \lambda a.\mathbb{S}_\beta[\mathbf{f}\ \mathbb{D}_\alpha[\lambda e.a], e] \tag{4.37c}$$

Equation (4.36a) says that if $\mathtt{a}$ is a term of type $(\gamma \to \iota)$, then the result of its dynamization is a itself (recall that $\bar{\iota}$ is $(\gamma \to \iota)$ and $\mathtt{a}$ is exactly of this type). According to Equation (4.36b), dynamization of a term P of type $(\gamma \to o)$

leads to a dynamic proposition (i.e. a term of type $(\gamma \to (\gamma \to o) \to o)$) and to the update of the context with the original static proposition. Compare (4.36b) with (4.25a). Equation (4.36c) specifies dynamization of a term f of the complex type $(\gamma \to (\alpha \to \beta))$. The resulting term has to be of type $(\overline{\alpha} \to \overline{\beta})$. It is defined by abstracting over a variable of type $\overline{\alpha}$ and specifying the body of type $\overline{\beta}$. The body has to be obtained by dynamization of a term of which f is a subterm. Since the body of the argument of the function $\mathbb{D}_\beta$ has to be the term of type β, f is given e and the static version $\mathbb{S}_\alpha[\mathbf{a}, e]$ of $\mathbf{a}$ as arguments. This can be more easily seen when types are explicitly presented, as shown below:

$$\mathbb{D}_{\alpha\to\beta}[f] \doteq \lambda \mathbf{a}^{\overline{\alpha}}.\, \mathbb{D}_\beta[\lambda e^\gamma.\, \overbrace{f^{\gamma\to(\alpha\to\beta)} e}^{\alpha\to\beta}\ \overbrace{\mathbb{S}_\alpha[\mathbf{a}, e]}^{\alpha}]$$

The reading function is the inverse of the dynamization function. It is a function of two arguments: a term to be transformed into its static equivalent and a context. Equation (4.37a) says that dynamic term of type $(\gamma \to \iota)$ is made static by giving the context as an argument to it. Equation (4.37b) shows that a dynamic proposition (of type $(\gamma \to (\gamma \to o) \to o)$) can be provided the context and an empty continuation to become a static proposition. Finally, Equation (4.37c) defines the translation of a dynamic term $\mathbf{f}$ of a complex type $(\overline{\alpha} \to \overline{\beta})$ to its static equivalent of type $(\alpha \to \beta)$. First, there is an abstraction over a variable of type α, then the body of type β is obtained by applying $\mathbb{S}_\beta$ to a term of which $\mathbf{f}$ is a subterm. Therefore, to return a term of type $\overline{\beta}$, $\mathbf{f}$ has to be provided a term of type $\overline{\alpha}$ as an argument. This term is obtained by applying dynamization function $\mathbb{D}_\alpha$ to $(\lambda e.\mathtt{a})$. This can be seen better when the types are explicitly shown, as below:

$$\mathbb{S}_{\alpha\to\beta}\,[\mathbf{f}, e] \doteq \lambda a^\alpha.\, \mathbb{S}_\beta[\mathbf{f}^{\overline{\alpha}\to\overline{\beta}}\ \overbrace{\mathbb{D}_\alpha[\lambda e.a]}^{\overline{\alpha}}, e]$$

Now, having specified dynamization of non-logical constants, the modular dynamization of any term can be defined:

Definition 4.28. [Dynamization of λ-terms] A term t of type δ is **dynamized** into a term $\overline{t}$ of type $\overline{\delta}$ in the following way:

- If t is a non-logical constant k, then

$$\overline{k} \doteq \mathbb{D}_\delta[\lambda e.k] \tag{4.38a}$$

- If t is a variable x, then

$$\overline{x} \doteq x \tag{4.38b}$$

- If t is an application vu, then

$$\overline{vu} \doteq \overline{v}\ \overline{u} \tag{4.38c}$$

- If t is an abstraction $\lambda x.v$, then

$$\overline{\lambda x.v} \doteq \lambda x.\overline{v} \tag{4.38d}$$

- If t is a conjunction, then

$$\overline{\wedge} \doteq \lambda \mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi) \tag{4.38e}$$

- If t is an existential quantifier, then

$$\overline{\exists} \doteq \lambda \mathbf{P}.\lambda e\phi.\exists(\lambda x.\mathbf{P}(\lambda e'.x)e\phi) \tag{4.38f}$$

- If t is a negation, then

$$\overline{\neg} \doteq \lambda \mathbf{A}.\lambda e\phi.\neg(\mathbf{A}e(\lambda e.\top)) \wedge \phi e \tag{4.38g}$$

A non-logical constant k is dynamized by applying function $\mathbb{D}$ to $(\lambda e.k)$, as defined in (4.38a). Dynamization of a variable is the same variable. Dynamization of an application is the application of each of the terms dynamized separately. Dynamization of an abstraction is the dynamization of the body of it, abstracted over the same variable.

As in (4.24b), dynamic conjunction (4.38e) is a composition of two terms in such a way that the second conjunct is part of the continuation of the first conjunct.

Dynamic existential quantifier (4.38f) simply introduces a new existentially quantified variable in the logical formula. Note that it does not update the environment with the introduced individual x, unlike the dynamic existential quantifier in (4.24c). This simplification is possible under the assumption that the environment has a richer structure, because (static) propositions in which x is a subterm are guaranteed to be appended into the environment due to Equation (4.36b) (i.e. when the proposition is dynamized) and, hence, x could be selected from the environment by applying a selection function that tries unification.

Dynamic negation (4.38f) is defined in a way that the continuation of the discourse does not fall in its scope. Therefore, the variables introduced under its scope are not available for binding in the future processing.

Analogously to Remark 4.22 abbreviations can be introduced in the dynamic case:

REMARK 4.29. Symbols $\overline{\vee}$, $\overline{\forall}$ and $\overline{\Rightarrow}$ are the dynamizations of the abbreviations given in 4.22:

$$\overline{\vee} \doteq \lambda\mathbf{AB}.\neg(\neg\mathbf{A}\overline{\wedge}\neg\mathbf{B})$$
$$\overline{\forall} \doteq \lambda\mathbf{P}.\neg(\overline{\exists}(\neg\mathbf{P}))$$
$$\overline{\Rightarrow} \doteq \lambda\mathbf{AB}.\neg(\mathbf{A}\overline{\wedge}\neg\overline{\mathbf{B}})$$

Note that symbols $\overline{\vee}$, $\overline{\forall}$ and $\overline{\Rightarrow}$ are simply abbreviations of complex terms where $\neg$, $\overline{\exists}$ and $\overline{\wedge}$ are the subterms. Particularly, each of these terms has a outermost negation $\neg$ and, therefore, the resulting environments are not updated with the contents of **A**, **B** or **P**.

4.3.2 Linguistic Motivations

Framework GL formally introduced in Subsection 4.3.1 offers several linguistically motivated innovations that make it more realistic than de Groote's original approach G_0 [de Groote, 2006], demonstrated in Section 4.1, and its refinement G, introduced in Section 4.2.

First of all, contexts have a more realistic structure and may contain some knowledge (including common ground). Taking, for example, referring expressions, this is necessary for retrieving an individual from a context based on the informational content of the expression. Consider, for instance, the expression *the man who smiles*. To select a referent for this expression, it is desirable to provide the property ($\lambda x.\mathbf{man}x \wedge \mathbf{smile}x$) and a context to the selection function sel as arguments. Thus, the selection function is defined as a function of type (4.40) that takes a property P and a context c and returns an individual $\mathtt{a}$ from c such that $\mathtt{a}$ satisfies P:

$$\mathsf{sel} : ((\iota \to o) \to \gamma \to \iota) \tag{4.40}$$

The context has to have an appropriate structure to allow the search and retrieval of a referent satisfying a descriptive content P. For illustrative purposes framework G_0 uses a simplified representation of the context as a list of individuals (i.e. $\gamma \doteq$ `list of` $[\iota]$). To demonstrate the elaborated framework GL based on continuation-based dynamic logic defined in Section 4.3.1, context can be viewed as a conjunction of formulas and γ can be assumed to be of the following type:

$$\gamma \doteq o \tag{4.41}$$

This gives an essential advance, namely the possibility of updating the context

with a proposition:[12]

$$\overline{\mathrm{P}t_1 \dots t_n} = \lambda e\phi.\mathrm{P}t_1 \dots t_n \wedge \phi\ \underbrace{(\mathsf{upd}(\mathrm{P}t_1 \dots t_n, e))}_{\substack{\text{update}\\ \text{context } e\\ \text{with proposition}\\ \mathrm{P}t_1 \dots t_n}} \tag{4.42}$$

Then, the notion of context containing a term (including an individual) can be defined in the following way:

DEFINITION 4.30. Context c **contains** term x (x **is stored in** c) if and only if x is a subterm of c.

Definition 4.31 below gives a possible formalization of sel when context is a conjunction of formulas. It specifies that, given context c and property P, sel returns an individual a stored in c satisfying the condition that c contains $\mathrm{P}\mathsf{a}$ (i.e. a has property P) and a is the only individual in c having property P:

DEFINITION 4.31. Let P be a term of type $(\iota \to o)$ and c be a term of type o. Then, $\mathsf{sel}\ \mathrm{P}\ c \doteq \mathsf{a}$ if and only if $c \vdash \mathrm{P}\mathsf{a}$ and, for all x, if $c \vdash \mathrm{P}x$ then $x = \mathsf{a}$.

EXAMPLE 4.32. Assume $\mathbf{non_human}, \mathbf{human}, \mathbf{male}, \mathbf{donkey}$ and $\mathbf{farmer}$ are constants of type $(\iota \to o)$. Then, the following equations hold:

$$\mathsf{sel}(\mathbf{non_human})(\mathbf{donkey}x \wedge \mathbf{non_human}x) = x$$
$$\mathsf{sel}(\lambda z.\mathbf{human}z \wedge \mathbf{male}z)(\mathbf{donkey}x \wedge \mathbf{farmer}y \wedge \mathbf{male}y \wedge \mathbf{human}y) = y$$

□

Context c contains knowledge c' that was recently learned from a preceding dialogue or discourse (e.g. *John loves Mary*) and common knowledge Σ (e.g. *There are twenty four hours in a day*).[13] Thus Definition 4.31 can be stated as follows:

DEFINITION 4.33. Let Σ be some theory formalizing (common) knowledge. Then, $\mathsf{sel}\ \mathrm{P}\ c' \doteq \mathsf{a}$ if and only if $\Sigma, c' \vdash \mathrm{P}\mathsf{a}$ and for all x if $\Sigma, c' \vdash \mathrm{P}x$, then $x = \mathsf{a}$.

Formalization of Σ is a difficult task.[14] In the scope of this dissertation Σ is represented by a finite set of formulas and, therefore, it can be expressed by a

[12] Compare (4.42) with (4.25a).

[13] The term "common knowledge" is due to [Lewis, 1969].

[14] The Stanford Encyclopedia's of Philosophy entry [Vanderschraaf and Sillari, 2009] on common knowledge contains a wide list of references to different approaches of analysing the concept of common knowledge and gives overviews of some of them. There are projects that aim at formalizing and implementing common knowledge. Cyc [Lenat, 1995], for example, is a logical reasoning system containing world's largest general-purpose knowledge base [Curtis *et al.*, 2005].

formula $\bigwedge_{\sigma\in\Sigma}\sigma$ of type o. Thus, by equating c with $(\bigwedge_{\sigma\in\Sigma}\sigma \wedge c')$ here, the context c is considered to contain the common knowledge and, therefore, it is sufficient to use Definition 4.31.

Function sel is only partially defined in 4.31. First, there may be no individual variable $\mathtt{a}$ in the context c such that $c \vdash \mathrm{P}\mathtt{a}$. This means that the context c fails with respect to familiarity presupposition and accommodation should take place. A possible way to handle presupposition accommodation is presented in Chapter 5. Second, the uniqueness requirement may not hold. This corresponds to the apparent ambiguity that can be resolved by a salience property of individuals in the context as it is proposed, for example, by Martin and Pollard [2010; 2011]; or by rhetorical relations that hold between segments of discourse. It can even be a real ambiguity which is often the case in natural language. The framework then has to cope with this ambiguity by "asking" for a clarification (as humans do). Resolving these apparent and real ambiguities is clearly an interesting and challenging problem that has to be solved for having a richer natural language interpretation framework. However, before that, a preliminary and fundamental work has to be done: mathematically clean and precise formalization of a compositional framework of semantics of natural language capable of incorporating results not only in the area of anaphora resolution, but also presupposition triggering and projection, modal subordination and other dynamic phenomena. De Groote's ideas show a promising direction towards such a framework as they were successfully applied in [de Groote and Lebedeva, 2010; Asher and Pogodalla, 2010a; Asher and Pogodalla, 2010b] and in Chapters 5, 6 and 7 of this dissertation. Thus, the simplified notion of context defined in 4.31 is a way to focus on the development of the compositional framework capable of dealing with certain (still complex) natural language phenomena. Subsequently, context can be enhanced to deal with even more complex natural languages phenomena.

Moreover, the static type of individuals ι is transformed by Definition 4.26 into type $(\gamma \to \iota)$ of dynamic individuals in the new framework. The necessity of this change can be illustrated by recalling the dynamic interpretation (4.10) of pronoun *he* from [de Groote, 2006], repeated below:

$$\lambda\mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}_{he}e)e\phi \tag{4.43}$$

Assuming that function sel is redefined as having type $((\iota \to o) \to \gamma \to \iota)$ (as motivated above), the interpretation of *he* in the new framework could be (4.44):

$$\lambda\mathbf{P}.\lambda e\phi.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)e\phi \tag{4.44}$$

Term (4.44) of type $((\overline{\iota} \to \overline{o}) \to \overline{o})$ is the dynamic counterpart of the type-raised[15] static interpretation. However, if there is a need to interpret a non-quantified noun phrase without type-raising, i.e. as simply having the type ι, the dynamic

[15] See Section 1.4.

term has to be of type $\bar{\iota}$. Then, considering that part of the meaning of a pronoun is the search for the referent based on the descriptive content of the pronoun, the desired interpretation would be (4.45), which is η-equivalent to (4.46):

$$\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x) \tag{4.45}$$

$$\lambda e.\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e \tag{4.46}$$

Since sel is of type $((\iota \to o) \to \gamma \to \iota)$ and $(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)$ is of type $(\iota \to o)$, terms (4.45) and (4.46) are of type $(\gamma \to \iota)$, which is, then, taken to be the dynamic version $\bar{\iota}$ of type ι.

A systematic modular translation of static terms into dynamic terms is provided by Definitions 4.27 and 4.28. A special non-trivial case is the translation of non-logical constants. Consider, for example, the non-logical constant $\mathbf{try}$ of type $(\iota \to (\iota \to o) \to o)$. Desirably, its dynamic version $\overline{\mathbf{try}}$ of type $(\bar{\iota} \to (\bar{\iota} \to \bar{o}) \to \bar{o})$ should be as follows, where ψ abbreviates $\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbf{P}(\lambda e'.y)e(\lambda e''.\top))$:

$$\lambda \mathbf{x}^{\bar{\iota}}\mathbf{P}^{\bar{\iota}\to\bar{o}}e^{\gamma}\phi^{\gamma\to o}.\underbrace{\mathbf{try}^{\iota\to(\iota\to o)\to o}(\overbrace{\mathbf{x}e}^{\iota})(\lambda y^{\iota}.\ \mathbf{P}(\overbrace{\lambda e'^{\gamma}.y}^{\gamma\to\iota})\ e\ (\overbrace{\lambda e''^{\gamma}.\top}^{\gamma\to o}))}_{\psi} \wedge \phi(\overbrace{\mathsf{upd}(\psi,e)}^{\gamma})^{o} \tag{4.47}$$

However, it would be problematic, if possible at all, to manually assign dynamic interpretations, as $\overline{\mathbf{try}}$ above, for all non-logical constants. Therefore, special systematic treatment is proposed in Definition 4.27. For example, by applying this definition, the desired dynamic interpretation (4.47) of $\mathbf{try}$ can be obtained:

Example 4.34. Consider the non-logical constant $\mathbf{try}$ of type $(\iota \to (\iota \to o) \to o)$. Its dynamic equivalent $\overline{\mathbf{try}}$ is computed as follows:

$$\begin{aligned}
&\overline{\mathbf{try}} \\
&= \mathbb{D}_{\iota\to(\iota\to o)\to o}[\lambda e.\mathbf{try}] \\
&= \lambda \mathbf{x}.\mathbb{D}_{(\iota\to o)\to o}[\lambda e.\mathbf{try}\mathbb{S}_{\iota}[\mathbf{x},e]] && \text{(by (4.36c))} \\
&= \lambda \mathbf{x}\mathbf{P}.\mathbb{D}_{o}[\lambda e.\mathbf{try}\mathbb{S}_{\iota}[\mathbf{x},e]\mathbb{S}_{\iota\to o}[\mathbf{P},e]] && \text{(by (4.36c))} \\
&= \lambda \mathbf{x}\mathbf{P}.\mathbb{D}_{o}[\lambda e.\mathbf{try}(\mathbf{x}e)\mathbb{S}_{\iota\to o}[\mathbf{P},e]] && \text{(by (4.37a))} \\
&= \lambda \mathbf{x}\mathbf{P}.\mathbb{D}_{o}[\lambda e.\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbb{S}_{o}[\mathbf{P}\mathbb{D}_{\iota}[\lambda e.y],e]] && \text{(by (4.37c))} \\
&= \lambda \mathbf{x}\mathbf{P}.\mathbb{D}_{o}[\lambda e.\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbb{S}_{o}[\mathbf{P}(\lambda e.y),e]] && \text{(by (4.36a))}
\end{aligned}$$

$$
\begin{aligned}
&= \lambda\mathbf{x}\mathbf{P}.\mathbb{D}_o[\lambda e.\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbf{P}(\lambda e.y)e(\lambda e.\top)] && \text{(by (4.37b))}\\
&= \lambda\mathbf{x}\mathbf{P}.\lambda e\phi.(\lambda e.\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbf{P}(\lambda e.y)e(\lambda e.\top)))e\wedge && \text{(by (4.36b))}\\
&\qquad\qquad \phi\ (\mathsf{upd}(\lambda e.\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbf{P}(\lambda e.y)e(\lambda e.\top)))e,e))\\
&\to_\beta \lambda\mathbf{x}\mathbf{P}.\lambda e\phi.\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbf{P}(\lambda e.y)e(\lambda e.\top))\wedge\\
&\qquad\qquad \phi\ (\mathsf{upd}(\mathbf{try}(\mathbf{x}e)(\lambda y.\mathbf{P}(\lambda e.y)e(\lambda e.\top)),e))
\end{aligned}
$$

□

A proposition analogous to 4.16 holds in GL:

Proposition 4.35. *For all terms* P *of type* $(\iota \to o)$, B *of type* o *and any variable* $\mathbf{x}$ *of type* $(\gamma \to \iota)$ *such that* $\mathbf{x} \notin FV(\overline{\mathrm{B}})$*, the following equivalence holds:*

$$\overline{\exists}(\lambda\mathbf{x}.\overline{\mathrm{P}}\mathbf{x})\ \overline{\wedge}\ \overline{\mathrm{B}} =_\beta \overline{\exists}(\lambda\mathbf{x}.\overline{\mathrm{P}}\mathbf{x}\ \overline{\wedge}\ \overline{\mathrm{B}})$$

Proof.

$$
\begin{aligned}
&\overline{\exists}(\lambda\mathbf{x}.\overline{\mathrm{P}}\mathbf{x})\ \overline{\wedge}\ \overline{\mathrm{B}}\\
\to^*_\beta\ &(\lambda e\phi.\exists(\lambda x.\mathrm{P}x\wedge\phi(\mathsf{upd}(\mathrm{P}x,e))))\ \overline{\wedge}\ \overline{\mathrm{B}} && \text{(by (4.38a), Def. 4.27 and (4.38f))}\\
=\ &\lambda e\phi.(\lambda e\phi.\exists(\lambda x.\mathrm{P}x\wedge\phi(\mathsf{upd}(\mathrm{P}x,e))))e(\lambda e.\overline{\mathrm{B}}e\phi) && \text{(by (4.38e))}\\
\to^*_\beta\ &\lambda e\phi.(\lambda e\phi.\exists(\lambda x.\mathrm{P}x\wedge\phi(\mathsf{upd}(\mathrm{P}x,e))))e(\lambda e.\mathrm{B}\wedge\phi(\mathsf{upd}(\mathrm{B},e)))\\
&&& \text{(by (4.38a) and Def. 4.27)}\\
\to^*_\beta\ &\lambda e\phi.\exists(\lambda x.\mathrm{P}x\wedge(\mathrm{B}\wedge\phi(\mathsf{upd}(\mathrm{B},\mathsf{upd}(\mathrm{P}x,e))))) && (4.48)
\end{aligned}
$$

$$
\begin{aligned}
&\overline{\exists}(\lambda\mathbf{x}.\overline{\mathrm{P}}\mathbf{x}\ \overline{\wedge}\ \overline{\mathrm{B}})\\
=\ &\overline{\exists}(\lambda\mathbf{x}.\lambda e\phi.(\overline{\mathrm{P}}\mathbf{x})e(\lambda e.\overline{\mathrm{B}}e\phi)) && \text{(by (4.38e))}\\
\to^*_\beta\ &\overline{\exists}(\lambda\mathbf{x}.\lambda e\phi.(\lambda e\phi.\mathrm{P}(\mathbf{x}e)\wedge\phi(\mathsf{upd}(\mathrm{P}(\mathbf{x}e),e)))e(\lambda e.\mathrm{B}\wedge\phi(\mathsf{upd}(\mathrm{B},e))))\\
&&& \text{(by (4.38a), and Def. 4.27)}\\
\to^*_\beta\ &\overline{\exists}(\lambda\mathbf{x}.\lambda e\phi.\mathrm{P}(\mathbf{x}e)\wedge(\mathrm{B}\wedge\phi(\mathsf{upd}(\mathrm{B},\mathsf{upd}(\mathrm{P}(\mathbf{x}e),e))))) && (4.49)\\
\to^*_\beta\ &\lambda e\phi.\exists(\lambda x.\mathrm{P}x\wedge(\mathrm{B}\wedge\phi(\mathsf{upd}(\mathrm{B},\mathsf{upd}(\mathrm{P}x,e))))) && \text{(by (4.38f))}
\end{aligned}
$$

□

4.3.3 Conservation

This section proves a conservation result for framework GL, analogous to Theorem 4.20 for framework G. Since framework GL is more elaborated, the proof is more complex and consists of several lemmas.

LEMMA 4.36. *For all terms t of type δ, terms $\mathbf{t}$ of type $\overline{\delta}$, terms e of type γ and any Γ, the following hold:*

1. *If $\Gamma \vdash t : \delta$, then $\Gamma \vdash \mathbb{D}_\delta[\lambda e.t] : \overline{\delta}$*
2. *If $\Gamma \vdash \mathbf{t} : \overline{\delta}$, then $e : \gamma, \Gamma \vdash \mathbb{S}_\delta[\mathbf{t}, e] : \delta$.*

Proof. The proof is by simultaneous induction on the type δ.

- If $\delta = \iota$, then $\overline{\delta} = \gamma \to \iota$.
  1. $\mathbb{D}_\delta[\lambda e.t] = \lambda e.t$. We have the following proof:

$$\dfrac{\overset{\varphi'}{\overline{\Gamma, e : \gamma \vdash t : \iota}}}{\Gamma \vdash \lambda e.t : \gamma \to \iota}\ \text{abs}$$

     By assumption, we have a proof φ of $\Gamma \vdash t : \iota$. We can construct φ' by adding $e : \gamma$ to the left side of every sequent in φ.
  2. $\mathbb{S}_\delta[\mathbf{t}, e] = \mathbf{t}e$. We have the following proof:

$$\dfrac{\overset{\varphi'}{\overline{e : \gamma, \Gamma \vdash \mathbf{t} : \gamma \to \iota}} \qquad e : \gamma, \Gamma \vdash e : \gamma}{e : \gamma, \Gamma \vdash \mathbf{t}e : \iota}\ \text{app}$$

     By assumption, we have a proof φ of $\Gamma \vdash \mathbf{t} : \gamma \to \iota$. We can construct φ' by adding $e : \gamma$ to the left side of every sequent in φ.

- If $\delta = o$, then $\overline{\delta} = \gamma \to (\gamma \to o) \to o$.
  1. $\mathbb{D}_\delta[\lambda e.t] = \lambda e\phi.t \wedge \phi(\mathsf{upd}(t, e))$. We have the following proof

$$\dfrac{\dfrac{\overset{\varphi'}{\overline{\Gamma, e : \gamma, \phi : \gamma \to o \vdash t : o}} \qquad \Gamma, e : \gamma, \phi : \gamma \to o \vdash \wedge : o \to o \to o \qquad \varphi_1}{\Gamma, e : \gamma, \phi : \gamma \to o \vdash t \wedge \phi(\mathsf{upd}(t, e)) : o}\ \text{app}}{\Gamma \vdash \lambda e\phi.t \wedge \phi(\mathsf{upd}(t, e)) : \gamma \to (\gamma \to o) \to o}\ \text{abs}$$

     where φ_1 is

$$\dfrac{\Gamma, e : \gamma, \phi : \gamma \to o \vdash \phi : \gamma \to o \qquad \Gamma, e : \gamma, \phi : \gamma \to o \vdash \mathsf{upd}(t, e) : \gamma}{\Gamma, e : \gamma, \phi : \gamma \to o \vdash \phi(\mathsf{upd}(t, e)) : o}\ \text{app}$$

     By assumption, we have a proof φ of $\Gamma \vdash t : o$. We can construct φ' by adding $e : \gamma, \phi : \gamma \to o$ to the left side of every sequent in φ.
  2. $\mathbb{S}_\delta[\mathbf{t}, e] = \mathbf{t}e(\lambda e.\top)$. We have the following proof:

$$\dfrac{\dfrac{\dfrac{\overset{\varphi'}{\vdots}}{e:\gamma,\Gamma\vdash \mathbf{t}:\gamma\to(\gamma\to o)\to o} \quad e:\gamma,\Gamma\vdash e:\gamma}{e:\gamma,\Gamma\vdash \mathbf{t}e:(\gamma\to o)\to o}\,\text{app} \quad \dfrac{e:\gamma,\Gamma,e':\gamma\vdash\top:o}{e:\gamma,\Gamma\vdash\lambda e'.\top:\gamma\to o}\,\text{abs}}{e:\gamma,\Gamma\vdash \mathbf{t}e(\lambda e'.\top):o}\,\text{app}$$

By assumption, we have a proof φ of $\Gamma\vdash \mathbf{t}:\gamma\to(\gamma\to o)\to o$. We can construct φ' by adding $e:\gamma$ to the left side of every sequent in φ.

- If $\delta=\alpha\to\beta$, then $\overline{\delta}=\overline{\alpha}\to\overline{\beta}$.

  1. $\mathbb{D}_\delta[\lambda e.t]=\lambda\mathbf{a}.\mathbb{D}_\beta[\lambda e.t\ \mathbb{S}_\alpha[\mathbf{a},e]]$. Then,

  $$\dfrac{\Gamma,\mathbf{a}:\overline{\alpha}\vdash\mathbb{D}_\beta[\lambda e.t\ \mathbb{S}_\alpha[\mathbf{a},e]]:\overline{\beta}}{\Gamma\vdash\lambda\mathbf{a}.\mathbb{D}_\beta[\lambda e.t\ \mathbb{S}_\alpha[\mathbf{a},e]]:\overline{\alpha}\to\overline{\beta}}\,\text{abs}$$

  Since, by I.H., for all t' of type β and all terms e of type γ, it holds that if $\Gamma\vdash t':\beta$, then $\Gamma\vdash\mathbb{D}_\beta[\lambda e.t']:\overline{\beta}$, we shall prove that $\Gamma\vdash\lambda e.t\ \mathbb{S}_\alpha[\mathbf{a},e]:\gamma\to\beta$. The proof is as follows:

  $$\dfrac{\dfrac{\overset{\varphi'}{\vdots}}{\Gamma,e:\gamma\vdash t:\alpha\to\beta} \quad \dfrac{\overset{\varphi''}{\vdots}}{\Gamma,e:\gamma\vdash\mathbb{S}_\alpha[\mathbf{a},e]:\alpha}\,\text{app}}{\dfrac{\Gamma,e:\gamma\vdash t\ \mathbb{S}_\alpha[\mathbf{a},e]:\beta}{\Gamma\vdash\lambda e.t\ \mathbb{S}_\alpha[\mathbf{a},e]:\gamma\to\beta}\,\text{abs}}$$

  By assumption, we have a proof φ of $\Gamma\vdash t:\alpha\to\beta$. We can construct φ' by adding $e:\gamma$ to the left side of every sequent in φ.

  The proof φ'' is given by Item 2 of the I.H. of this lemma.

  2. $\mathbb{S}_\delta[\mathbf{t},e]=\lambda\mathtt{a}.\mathbb{S}_\beta[\mathbf{t}\ \mathbb{D}_\alpha[\lambda e'.\mathtt{a}],e]$. Then,

  $$\dfrac{e:\gamma,\Gamma,\mathtt{a}:\alpha\vdash\mathbb{S}_\beta[\mathbf{t}\ \mathbb{D}_\alpha[\lambda e'.\mathtt{a}],e]:\beta}{e:\gamma,\Gamma\vdash\lambda\mathtt{a}.\mathbb{S}_\beta[\mathbf{t}\ \mathbb{D}_\alpha[\lambda e'.\mathtt{a}],e]:\alpha\to\beta}\,\text{abs}$$

  Since, by I.H., for all $\mathbf{t}'$ of type $\overline{\beta}$ and all terms e of type γ it holds that if $\Gamma\vdash\mathbf{t}':\overline{\beta}$, then $e:\gamma,\Gamma\vdash\mathbb{S}_\beta[\mathbf{t}',e]:\beta$, we shall prove that $\Gamma\vdash\mathbf{t}\ \mathbb{D}_\alpha[\lambda e'.\mathtt{a}]:\overline{\beta}$. The proof is as follows:

  $$\dfrac{\dfrac{\overset{\varphi}{\vdots}}{\Gamma\vdash\mathbf{t}:\overline{\alpha}\to\overline{\beta}} \quad \dfrac{\overset{\varphi''}{\vdots}}{\Gamma\vdash\mathbb{D}_\alpha[\lambda e'.\mathtt{a}]:\overline{\alpha}}\,\text{app}}{\Gamma\vdash\mathbf{t}\ \mathbb{D}_\alpha[\lambda e'.\mathtt{a}]:\overline{\beta}}\,\text{app}$$

  The proof φ of $\Gamma\vdash\mathbf{t}:\overline{\alpha}\to\overline{\beta}$ is given by assumption.

  The proof φ'' is given by Item 1 of the I.H. of this lemma.

□

LEMMA 4.37. *Let t be a logical constant of type δ. Then for any Γ, $\overline{\Gamma} \vdash \overline{t} : \overline{\delta}$.*

Proof.

- If t is the logical conjunction $\wedge$, then $\delta = o \to o \to o$. We have to prove that there is a derivation φ' of $\overline{\Gamma} \vdash \overline{\wedge} : \overline{o \to o \to o}$. By Definition 4.25, there is an axiom φ of $\Gamma \vdash \wedge : o \to o \to o$. Using Equation (4.38e) of Definition 4.28 and Equation (4.35c) of Definition 4.26, the proof φ' is as shown in Figure 4.3.

- If t is the logical existential quantifier $\exists$, then $\delta = (\iota \to o) \to o$. We have to prove that there is a derivation φ' of $\overline{\Gamma} \vdash \overline{\exists} : \overline{(\iota \to o) \to o}$. By Definition 4.25, there is an axiom φ of $\Gamma \vdash \exists : (\iota \to o) \to o$. Using Equation (4.38f) of Definition 4.28 and Equation (4.35c) of Definition 4.26, the proof φ' is as shown in Figure 4.4.

- If t is the logical negation $\neg$, then $\delta = o \to o$. We have to prove that there is a derivation φ' of $\overline{\Gamma} \vdash \overline{\neg} : \overline{o \to o}$. By Definition 4.25, there is an axiom φ of $\Gamma \vdash \neg : o \to o$. Using Equation (4.38g) of Definition 4.28 and Equation (4.35c) of Definition 4.26, the proof φ' is as shown in Figure 4.5.

$$\dfrac{\dfrac{\dfrac{\varphi_1 \qquad \varphi_2}{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \mathbf{A}e(\lambda e'.\mathbf{B}e'\phi) : o}\,\text{app}}{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o} \vdash \lambda e\phi.\mathbf{A}e(\lambda e'.\mathbf{B}e'\phi) : \gamma \to (\gamma \to o) \to o}\,\text{abs}}{\overline{\Gamma} \vdash \lambda \mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e'.\mathbf{B}e'\phi) : \overline{o} \to \overline{o} \to \overline{o}}\,\text{abs}$$

where φ_1 is

$$\dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \mathbf{A} : \gamma \to (\gamma \to o) \to o \qquad \overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash e : \gamma}{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \mathbf{A}e : (\gamma \to o) \to o}\,\text{app}$$

and φ_2 is

$$\dfrac{\dfrac{\dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o, e' : \gamma \vdash \mathbf{B} : \gamma \to (\gamma \to o) \to o \qquad \overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o, e' : \gamma \vdash e' : \gamma}{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o, e' : \gamma \vdash \mathbf{B}e' : (\gamma \to o) \to o}\,\text{app} \qquad \overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o, e' : \gamma \vdash \phi : \gamma \to o}{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o, e' : \gamma \vdash \mathbf{B}e'\phi : o}\,\text{app}}{\overline{\Gamma}, \mathbf{A} : \overline{o}, \mathbf{B} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \lambda e'.\mathbf{B}e'\phi : \gamma \to o}\,\text{abs}$$

Figure 4.3: Proof of $\overline{\Gamma} \vdash \overline{\wedge} : \overline{o \to o \to o}$

$$\dfrac{\dfrac{\dfrac{

$$\dfrac{\varphi_1 \qquad \overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \wedge : o \to o \to o \qquad \dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \phi : \gamma \to o \qquad \overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash e : \gamma}{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \phi e : o}\ \text{app}}{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \neg(\mathbf{A} e (\lambda e'.\top)) \wedge \phi e : o}\ \text{app}}{\dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o} \vdash \lambda e \phi.\neg(\mathbf{A} e (\lambda e'.\top)) \wedge \phi e : \gamma \to (\gamma \to o) \to o}{\overline{\Gamma} \vdash \lambda \mathbf{A}.\lambda e \phi.\neg(\mathbf{A} e (\lambda e'.\top)) \wedge \phi e : \overline{o} \to \overline{o}}\ \text{abs}}\ \text{abs}$$

where φ_1 is

$$\dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \neg : o \to o \qquad \dfrac{\dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \mathbf{A} : \gamma \to (\gamma \to o) \to o \qquad \overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash e : \gamma}{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \mathbf{A} e : (\gamma \to o) \to o}\ \text{app} \qquad \dfrac{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o, e' : \gamma \vdash \top : o}{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \lambda e'.\top : \gamma \to o}\ \text{abs}}{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \mathbf{A} e (\lambda e'.\top) : o}\ \text{app}}{\overline{\Gamma}, \mathbf{A} : \overline{o}, e : \gamma, \phi : \gamma \to o \vdash \neg(\mathbf{A} e (\lambda e'.\top)) : o}\ \text{app}$$

Figure 4.5: Proof of $\overline{\Gamma} \vdash \neg : \overline{o \to o}$

□

Proposition 4.38. *For any term t of type δ and any Γ, if $\Gamma \vdash t : \delta$, then $\overline{\Gamma} \vdash \overline{t} : \overline{\delta}$.*

Proof. The proof is by induction on the structure of the term t.

- t is a variable x.

  By assumption, there is a derivation φ of $\Gamma \vdash x : \delta$, and we have to prove that there is a derivation φ' of $\overline{\Gamma} \vdash \overline{x} : \overline{\delta}$.

  Γ is necessarily of the form $\Gamma', x : \delta$. Therefore, $\overline{\Gamma}$ is $\overline{\Gamma'}, \overline{x} : \overline{\delta}$. The proof is just a single axiom:

$$\frac{}{\overline{\Gamma'}, \overline{x} : \overline{\delta} \vdash \overline{x} : \overline{\delta}}$$

- t is a non-logical constant k.

  By assumption, there is a derivation of $\Gamma \vdash k : \delta$, and we have to prove that there is a derivation of $\overline{\Gamma} \vdash \overline{k} : \overline{\delta}$.

  By Equation (4.38a) of Definition 4.28, $\overline{k} = \mathbb{D}_\delta\,[\lambda e.k]$. Therefore, there exists a derivation of $\Gamma \vdash \mathbb{D}_\delta[\lambda e.t] : \overline{\delta}$ by Lemma 4.36.

- t is a logical constant c, $\overline{\Gamma} \vdash \overline{c} : \overline{\delta}$ holds by Lemma 4.37.

- t is an application vu.

  By assumption, there is a derivation φ of $\Gamma \vdash vu : \beta$, and we have to prove that there is a derivation φ' of $\overline{\Gamma} \vdash \overline{vu} : \overline{\beta}$. By the typing rules, φ is necessarily of the form

$$\frac{\overset{\varphi_1}{\overline{\Gamma \vdash v : \alpha \to \beta}} \qquad \overset{\varphi_2}{\overline{\Gamma \vdash u : \alpha}}}{\Gamma \vdash vu : \beta}\ \text{app}$$

  By I.H. there are derivations φ'_1 and φ'_2 of $\overline{\Gamma} \vdash \overline{v} : \overline{\alpha \to \beta}$ and $\overline{\Gamma} \vdash \overline{u} : \overline{\alpha}$ respectively.

  By Equation (4.35c) of Definition 4.26, φ'_1 is a derivation of $\overline{\Gamma} \vdash \overline{v} : \overline{\alpha} \to \overline{\beta}$. Then, we can construct the following proof:

$$\frac{\overset{\varphi'_1}{\overline{\overline{\Gamma} \vdash \overline{v} : \overline{\alpha} \to \overline{\beta}}} \qquad \overset{\varphi'_2}{\overline{\overline{\Gamma} \vdash \overline{u} : \overline{\alpha}}}}{\overline{\Gamma} \vdash \overline{v}\ \overline{u} : \overline{\beta}}\ \text{app}$$

By Equation (4.38c) of Definition 4.28, the proof above is a derivation φ' of $\overline{\Gamma} \vdash \overline{vu} : \overline{\beta}$, as desired.

- t is an abstraction $\lambda x.v$.

  By assumption, there is a derivation φ of $\Gamma \vdash \lambda x.v : \alpha \to \beta$, and we have to prove that there is a derivation φ' of $\overline{\Gamma} \vdash \overline{\lambda x.v} : \overline{\alpha \to \beta}$.

  By the typing rules, φ is necessarily of the form

$$\dfrac{\begin{matrix}\\ \Gamma, x : \alpha \vdash v : \beta\end{matrix}\ \varphi_1}{\Gamma \vdash \lambda x.v : \alpha \to \beta}\ \text{abs}$$

  By I.H. there is a derivation φ'_1 of $\overline{\Gamma}, \overline{x} : \overline{\alpha} \vdash \overline{v} : \overline{\beta}$. Then, we can construct the following proof:

$$\dfrac{\begin{matrix}\\ \overline{\Gamma}, \overline{x} : \overline{\alpha} \vdash \overline{v} : \overline{\beta}\end{matrix}\ \varphi'_1}{\overline{\Gamma} \vdash \lambda \overline{x}.\overline{v} : \overline{\alpha} \to \overline{\beta}}\ \text{abs}$$

  By Equations (4.38d) and (4.35c) of Definitions 4.28 and 4.26 respectively, the proof above is a derivation φ' of $\overline{\Gamma} \vdash \overline{\lambda x.v} : \overline{\alpha \to \beta}$, as desired.

□

LEMMA 4.39. [Substitution] *For all terms t and w, $\overline{t}[x := \overline{w}] = \overline{t[x := w]}$.*

Proof. The proof is by induction on the structure of t.

- if t is a variable x, then

$$\begin{aligned} \overline{x}[x := \overline{w}] &= x[x := \overline{w}] && \text{(by (4.38b))} \\ &= \overline{w} \\ &= \overline{x[x := w]} \end{aligned}$$

- if t is a non-logical constant k, then

$$\begin{aligned} \overline{k}[x := \overline{w}] &= \overline{k} \\ &= \overline{k[x := w]} \end{aligned}$$

- if t is an application vu, then

$$\begin{aligned}
\overline{vu}[x := \overline{w}] &= (\overline{vu})[x := \overline{w}] && \text{(by (4.38c))}\\
&= \overline{v}[x := \overline{w}]\ \overline{u}[x := \overline{w}]\\
&= \overline{v[x := w]}\ \overline{u[x := w]} && \text{(by I.H.)}\\
&= \overline{v[x := w]\ u[x := w]} && \text{(by (4.38c))}\\
&= \overline{vu[x := w]}
\end{aligned}$$

- if t is an abstraction $\lambda x.v$, then

$$\begin{aligned}
(\overline{\lambda y.v})[x := \overline{w}] &= (\lambda y.\overline{v})[x := \overline{w}] && \text{(by (4.38d))}\\
&= \lambda y.(\overline{v}[x := \overline{w}])\\
&= \lambda y.(\overline{v[x := w]}) && \text{(by I.H.)}\\
&= \overline{\lambda y.(v[x := w])} && \text{(by (4.38d))}\\
&= \overline{(\lambda y.v)[x := w]}
\end{aligned}$$

□

Lemma 4.40. *For all terms v and u, such that $v = u$, $\overline{v} = \overline{u}$.*

Proof. Since $v = u$ (by assumption), there is a normal form w, such that $v \to_\beta^* w$ and $u \to_\beta^* w$. The proof is by induction on the length of the β-normalization of v.

- v is in normal form. The proof continues by induction on the length of the β-normalization of u.
  - If u is in normal form, then v and u are syntactically equal. Hence, $\overline{v} = \overline{u}$ holds.
  - Assume by I.H. that the lemma holds for $v =_s w$ and for all u' such that $u' \to_\beta^n w$. We need to show that it also holds for $u \to_\beta^{n+1} w$. Since $u' \to_\beta^n w$ and $v =_s w$, $u' \to_\beta^n v$. Then, by definition of equality, $u' = v$ and, by I.H., $\overline{u'} = \overline{v}$. Therefore, to show that $\overline{u}$ is equal to $\overline{v}$, it sufficies to show that $\overline{u}$ is equal to $\overline{u'}$. Since u is not in normal form, and u' is obtained from u in one β-reduction step, u and u' must be of the following forms: $u = t[y := (\lambda x.h)t']$ and $u' = t[y := h[x := t']]$. By Lemma 4.39 and Definition 4.28, $\overline{u} = \overline{t}[y := (\lambda x.\overline{h})\overline{t'}]$ and $\overline{u'} = \overline{t}[y := \overline{h}[x := \overline{t}']]$. Then, $\overline{u} \to_\beta \overline{u'}$. Therefore, $\overline{u} = \overline{u'}$.
- Assume by I.H. that the lemma holds for all v', such that $v' \to_\beta^m w$. We need to show that it also holds for $v \to_\beta^{m+1} w$. The proof proceeds by induction on the length of the β-normalization of u.

- If u is in normal form, then we have $v' \rightarrow_\beta^n u$. Then, by definition of equality, $v' = u$ and, by I.H., $\overline{v'} = \overline{u}$. Therefore, to show that $\overline{v}$ is equal to $\overline{u}$, it sufficies to show that $\overline{v}$ is equal to $\overline{v'}$. Since v is not in normal form, and v' is obtained from v in one β-reduction step, v and v' must be of the following forms: $v = t[y := (\lambda x.h)t']$ and $v' = t[y := h[x := t']]$. By Lemma 4.39 and Definition 4.28, $\overline{v} = \overline{t}[y := (\lambda x.\overline{h})\overline{t'}]$ and $\overline{v'} = \overline{t}[y := \overline{h}[x := \overline{t}']]$. Then, $\overline{v} \rightarrow_\beta \overline{v'}$. Therefore, $\overline{v} = \overline{v'}$.
- Assume by I.H. that the lemma holds for all u' such that $u' \rightarrow_\beta^n w$. We need to show that it also holds for $u \rightarrow_\beta^{n+1} w$. Since $v = u$ (by assumption), $v \rightarrow_\beta v'$ and $u \rightarrow_\beta u'$, $v' = u'$. Then, by I.H. $\overline{v'} = \overline{u'}$. Therefore, to prove that $\overline{v} = \overline{u}$, it sufficies to show that $\overline{v} = \overline{v'}$ and $\overline{u} = \overline{u'}$. Since v is not in normal form, and v' is obtained from v in one β-reduction step, v and v' must be of the following forms:$v = t[y := (\lambda x.h)t']$ and $v' = t[y := h[x := t']]$. By Lemma 4.39 and Definition 4.28, $\overline{v} = \overline{t}[y := (\lambda x.\overline{h})\overline{t'}]$ and $\overline{v'} = \overline{t}[y := \overline{h}[x := \overline{t}']]$. Then, $\overline{v} \rightarrow_\beta \overline{v'}$, and, therefore, $\overline{v} = \overline{v'}$. Since u is not in normal form, and u' is obtained from u in one β-reduction step, u and u' must be of the following forms: $u = t[y := (\lambda x.h)t']$ and $u' = t[y := h[x := t']]$. By Lemma 4.39 and Definition 4.28, $\overline{u} = \overline{t}[y := (\lambda x.\overline{h})\overline{t'}]$ and $\overline{u'} = \overline{t}[y := \overline{h}[x := \overline{t}']]$. Then, $\overline{u} \rightarrow_\beta \overline{u'}$. Therefore, $\overline{u} = \overline{u'}$.

□

NOTATION 4.41. Let t be a term of type δ and let $FV(t)$ be the set $\{x_1, \ldots, x_n\}$ of types $\{\alpha_1, \ldots, \alpha_n\}$. $\overline{t}^*$ denotes $\overline{t}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]]$.

LEMMA 4.42. *For all terms v and u of types $\alpha \rightarrow \beta$ and α respectively, $\overline{vu}^* = \overline{v}^*\ \overline{u}^*$.*

Proof.

$$
\begin{aligned}
\overline{vu}^* &= \overline{vu}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]] && \text{(by Notation 4.41)}\\
&= (\overline{v}\ \overline{u})[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]] && \text{(by (4.38c))}\\
&= \overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]]\\
&\qquad\qquad \overline{u}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]]\\
&= \overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_m := \mathbb{D}_{\alpha_m}[\lambda e.x_m]]\\
&\qquad\qquad \overline{u}[x_{m+1} := \mathbb{D}_{\alpha_{m+1}}[\lambda e.x_{m+1}], \ldots, x_k := \mathbb{D}_{\alpha_k}[\lambda e.x_k]]\\
&= \overline{v}^*\ \overline{u}^*
\end{aligned}
$$

□

LEMMA 4.43. *For all terms v and u, such that $v = u$, $\overline{v}^* = \overline{u}^*$.*

Proof. By assumption $v = u$. Therefore, by Lemma 4.40, $\overline{v} = \overline{u}$. Then, by the substitution lemma A.28 the following holds:

$$\begin{aligned}&\overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]]\\ =\ &\overline{u}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]]\end{aligned}$$

Hence, $\overline{v}^* = \overline{u}^*$. □

PROPOSITION 4.44. *For any term t of type δ and any Γ, if $\Gamma \vdash t : \delta$, then $\Gamma \vdash \overline{t}^* : \overline{\delta}$.*

Proof. The proof is by induction on the structure of the term t.

- t is a variable x.

  By assumption, there is a derivation of $\Gamma \vdash x : \delta$, and we have to prove that there is a derivation of $\Gamma \vdash \overline{x}^* : \overline{\delta}$.

  By Lemma 4.36, there exists a derivation of $\Gamma \vdash \mathbb{D}_\delta[\lambda e.x] : \overline{\delta}$. By Notation 4.41 and Equation (4.38b) of Definition 4.28, $\overline{x}^*$ is equal to $\mathbb{D}_\delta[\lambda e.x]$. Hence, $\Gamma \vdash \overline{x}^* : \overline{\delta}$.

- t is a non-logical constant k.

  By assumption, there is a derivation of $\Gamma \vdash k : \delta$, and we have to prove that there is a derivation of $\Gamma \vdash \overline{k}^* : \overline{\delta}$.

  By Lemma 4.36, there exists a derivation of $\Gamma \vdash \mathbb{D}_\delta[\lambda e.k] : \overline{\delta}$. By Notation 4.41 and Equation (4.38a) of Definition 4.28, $\overline{k}^*$ is equal to $\mathbb{D}_\delta[\lambda e.k]$. Hence, $\Gamma \vdash \overline{k}^* : \overline{\delta}$.

- t is a logical constant k (i.e. $\wedge$, $\exists$ or $\neg$).

  We have to prove that there is a derivation of $\Gamma \vdash \overline{k}^* : \overline{\delta}$. Definition 4.25 provides axioms of the form $\Gamma \vdash k : \delta$. Then, by Lemma 4.37, $\Gamma \vdash \overline{k} : \overline{\delta}$. Consequently, since $\overline{k} = \overline{k}^*$, $\Gamma \vdash \overline{k}^* : \overline{\delta}$.

- t is an application vu.

  By assumption, there is a derivation φ of $\Gamma \vdash vu : \beta$, and we have to prove that there is a derivation φ' of $\Gamma \vdash \overline{vu}^* : \overline{\beta}$. Applying Notation 4.41, Lemma 4.42 and Equation (4.35c) of Definition 4.26, the proof φ' is as follows:

$$\dfrac{\Gamma \vdash \overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]] : \overline{\alpha} \to \overline{\beta} \;[\varphi'_1] \qquad \Gamma \vdash \overline{u}[y_1 := \mathbb{D}_{\alpha'_1}[\lambda e.y_1], \ldots, y_m := \mathbb{D}_{\alpha'_m}[\lambda e.y_m]] : \overline{\beta} \;[\varphi'_2]}{\Gamma \vdash \overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]]\ \overline{u}[y_1 := \mathbb{D}_{\alpha'_1}[\lambda e.y_1], \ldots, y_m := \mathbb{D}_{\alpha'_m}[\lambda e.y_m]] : \overline{\beta}}\ \text{app}$$

By the typing rules, φ is necessarily of the form

$$\dfrac{\overset{\varphi_1}{\Gamma \vdash v : \alpha \to \beta} \qquad \overset{\varphi_2}{\Gamma \vdash u : \alpha}}{\Gamma \vdash vu : \beta}\ \text{app}$$

Then, by I.H., there are derivations φ_1' and φ_2' of $\Gamma \vdash \overline{v}^* : \overline{\alpha \to \beta}$ and $\Gamma \vdash \overline{u}^* : \overline{\alpha}$ respectively.

- t is an abstraction $\lambda x.v$.

  By assumption, there is a derivation φ of $\Gamma \vdash \lambda x.v : \alpha \to \beta$, and we have to prove that there is a derivation φ' of $\Gamma \vdash \overline{\lambda x.v}^* : \overline{\alpha \to \beta}$. Applying Notation 4.41 and Definitions 4.38d and 4.35c, the proof φ' is as follows:

$$\dfrac{\overset{\varphi''}{\Gamma, x : \overline{\alpha} \vdash \overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]] : \overline{\beta}}}{\Gamma \vdash \lambda x.\overline{v}[x_1 := \mathbb{D}_{\alpha_1}[\lambda e.x_1], \ldots, x_n := \mathbb{D}_{\alpha_n}[\lambda e.x_n]] : \overline{\alpha} \to \overline{\beta}}\ \text{abs}$$

  By the typing rules, φ is necessarily of the form

$$\dfrac{\overset{\varphi'''}{\Gamma, x : \alpha \vdash v : \beta}}{\Gamma \vdash \lambda x.v : \alpha \to \beta}\ \text{abs}$$

  Then, by I.H., there is a derivation φ'''' of $\Gamma, x : \alpha \vdash \overline{v}^* : \overline{\beta}$.

  The proof φ'' can be constructed from the proof φ'''' by substituting all occurences of subproof of $\Gamma, x : \alpha, \Gamma' \vdash \mathbb{D}_\alpha[\lambda e.x] : \overline{\alpha}$ in φ'''' with the axiom $\Gamma, x : \overline{\alpha}, \Gamma' \vdash x : \overline{\alpha}$ and propagating the changes down, i.e. substituting all $\mathbb{D}_\alpha[\lambda e.x] : \overline{\alpha}$ by $x : \overline{\alpha}$ on the right side and $x : \alpha$ by $x : \overline{\alpha}$ on the left side of the sequents occuring in the same branch of the replaced axiom.

□

Definition 4.45. [Stable term] A term t of type δ is said to be **stable** if and only if, for all terms u of type γ, $\mathbb{S}_\delta[\overline{t}^*, u] = t$.

Definition 4.46. [Reducible terms] The family of sets Red_δ of **reducible terms** is inductively defined as follows:

- $t \in \mathsf{Red}_\iota$ if and only if, for all terms u of type γ, $\overline{t}^* u = t$.

- $t \in \mathsf{Red}_o$ if and only if, for all terms u and u' of type γ, and for all terms v of type o such that $u' \notin FV(v)$, $\overline{t}^* u(\lambda u'.v) = t \wedge v$.
- $t \in \mathsf{Red}_{\alpha\to\beta}$ if and only if, for all terms $u \in \mathsf{Red}_\alpha$, $tu \in \mathsf{Red}_\beta$.

PROPOSITION 4.47. *Every term is stable.*

Proof. It is necessary to prove that

1. every term is reducible
2. every reducible term is stable

The proof consists of Lemmas 4.50, 4.52, 4.53, 4.55, as well as Corollaries 4.51 and 4.54, and Remark 4.48.

REMARK 4.48. By Definition 4.46, for all t and u, if $t \in \mathsf{Red}_{\alpha\to\beta}$ and $u \in \mathsf{Red}_\alpha$, then $tu \in \mathsf{Red}_\beta$.

LEMMA 4.49. *Let h be a variable or a non-logical constant of type $\alpha_1 \to \cdots \to \alpha_n \to \delta$, and let $t_1, \ldots, t_n$ be stable terms of types $\alpha_1, \ldots, \alpha_n$ respectively. Then, the following equation hold: $\overline{ht_1 \ldots t_n}^* = \mathbb{D}_\delta[\lambda e.h\ t_1 \ldots t_n]$*

Proof. Let e be a term of type γ. If h is a variable, then

$$\begin{aligned}
\overline{ht_1 \ldots t_n}^* &= \overline{h}^* \overline{t_1}^* \ldots \overline{t_n}^* && \text{(by 4.42)}\\
&= \overline{h}[h := \mathbb{D}_{\alpha_1\to\cdots\to\alpha_n\to\delta}[\lambda e.h]]\overline{t_1}^* \ldots \overline{t_n}^* && \text{(by 4.41)}\\
&= h[h := \mathbb{D}_{\alpha_1\to\cdots\to\alpha_n\to\delta}[\lambda e.h]]\overline{t_1}^* \ldots \overline{t_n}^* && \text{(by (4.38b))}\\
&= \mathbb{D}_{\alpha_1\to\cdots\to\alpha_n\to\delta}[\lambda e.h]\overline{t_1}^* \ldots \overline{t_n}^*\\
&= (\lambda \mathbf{a}_1 \ldots \mathbf{a}_n.\mathbb{D}_\delta[\lambda e.h\ \mathbb{S}_{\alpha_1}[\mathbf{a}_1, e] \ldots \mathbb{S}_{\alpha_n}[\mathbf{a}_n, e]])\overline{t_1}^* \ldots \overline{t_n}^* && \text{(by (4.36c))}\\
&\to_\beta^* \mathbb{D}_\delta[\lambda e.h\ \mathbb{S}_{\alpha_1}[\overline{t_1}^*, e] \ldots \mathbb{S}_{\alpha_n}[\overline{t_n}^*, e]]\\
&= \mathbb{D}_\delta[\lambda e.h\ t_1 \ldots t_n] && \text{(since } t_1 \ldots t_n \text{ are stable)}
\end{aligned}$$

If h is a non-logical constant, then

$$\begin{aligned}
\overline{ht_1 \ldots t_n}^* &= \overline{h}^* \overline{t_1}^* \ldots \overline{t_n}^* && \text{(by 4.42)}\\
&= \overline{h}\overline{t_1}^* \ldots \overline{t_n}^* && \text{(by 4.41)}\\
&= \mathbb{D}_{\alpha_1\to\cdots\to\alpha_n\to\delta}[\lambda e.h]\overline{t_1}^* \ldots \overline{t_n}^* && \text{(by (4.38a))}\\
&= (\lambda \mathbf{a}_1 \ldots \mathbf{a}_n.\mathbb{D}_\delta[\lambda e.h\ \mathbb{S}_{\alpha_1}[\mathbf{a}_1, e] \ldots \mathbb{S}_{\alpha_n}[\mathbf{a}_n, e]])\overline{t_1}^* \ldots \overline{t_n}^* && \text{(by (4.36c))}\\
&\to_\beta^* \mathbb{D}_\delta[\lambda e.h\ \mathbb{S}_{\alpha_1}[\overline{t_1}^*, e] \ldots \mathbb{S}_{\alpha_n}[\overline{t_n}^*, e]]\\
&= \mathbb{D}_\delta[\lambda e.h\ t_1 \ldots t_n] && \text{(since } t_1 \ldots t_n \text{ are stable)}
\end{aligned}$$

□

LEMMA 4.50.

1. *Let h be a variable or a non-logical constant, and let $t_1, \ldots, t_n$ be stable terms. Then, the term $ht_1, \ldots, t_n$ of type δ is reducible.*

2. *Every reducible term of type δ is stable modulo logical equivalence.*

Proof.

1. and 2. are proven simultaneously by induction on the type δ.

- $\delta = \iota$

  1. Let u be a term of type γ. By Definition 4.46, $\overline{ht_1 \ldots t_n}^* u = h\ t_1 \ldots t_n$ should hold. Indeed,

$$\begin{aligned} \overline{ht_1 \ldots t_n}^* u &= \mathbb{D}_\iota[\lambda e.h\ t_1 \ldots t_n]u && \text{(by 4.49)} \\ &= (\lambda e.h\ t_1 \ldots t_n)u && \text{(by (4.36a))} \\ &\to_\beta h\ t_1 \ldots t_n \end{aligned}$$

  2. Let t be a reducible term of type ι, v be a term of type γ. Then, by Definition 4.45, $\mathbb{S}_\iota[\overline{t}^*, v] = t$ should hold. Indeed,

$$\begin{aligned} \mathbb{S}_\iota[\overline{t}^*, v] &= \overline{t}^* v && \text{(by (4.37a))} \\ &= t && \text{(since } t \in \mathsf{Red}_\iota\text{)} \end{aligned}$$

- $\delta = o$

  1. Let u, u' be terms of type γ and v be a term of type o, such that $u' \notin FV(v)$. By Definition 4.46, $\overline{ht_1 \ldots t_n}^* u(\lambda u'.v) = h\ t_1 \ldots t_n \wedge v$ should hold. Indeed,

$$\begin{aligned} &\overline{ht_1 \ldots t_n}^* u(\lambda u'.v) \\ &= \mathbb{D}_o[\lambda e.h\ t_1 \ldots t_n]u(\lambda u'.v) && \text{(by 4.49)} \\ &= (\lambda e\phi.h\ t_1 \ldots t_n \wedge \phi(\mathsf{upd}(h\ t_1 \ldots t_n, e)))u(\lambda u'.v) && \text{(by (4.36b))} \\ &\to_\beta (\lambda \phi.h\ t_1 \ldots t_n \wedge \phi(\mathsf{upd}(h\ t_1 \ldots t_n, u)))(\lambda u'.v) \\ &\to_\beta h\ t_1 \ldots t_n \wedge (\lambda u'.v)(\mathsf{upd}(h\ t_1 \ldots t_n, u)) \\ &\to_\beta h\ t_1 \ldots t_n \wedge v && \text{(since } u' \notin FV(v)\text{)} \end{aligned}$$

  2. Let t be a reducible term of type o, v and v' be terms of type γ. Then, by Definition 4.45, $\mathbb{S}_o[\overline{t}^*, v] = t$ should hold. Indeed,

$$\begin{aligned} \mathbb{S}_o[\overline{t}^*, v] &= \overline{t}^* v(\lambda v'.\top) && \text{(by (4.37b))} \\ &= t \wedge \top && \text{(since } t \in \mathsf{Red}_o\text{)} \\ &\equiv t \end{aligned}$$

- $\delta = \alpha \to \beta$

  1. Let u be a reducible term of type α. By I.H.2. u is stable. Then, by I.H.1. $ht_1 \dots t_n u$ is reducible. Then, by Definition 4.46 $ht_1 \dots t_n$ is also reducible.
  2. Let u be a term of type γ. By Definition 4.45, $\mathbb{S}_{\alpha\to\beta}[\overline{ht_1 \dots t_n}^*, u] = ht_1 \dots t_n$ should hold. Indeed,

$$\begin{aligned}
\mathbb{S}_{\alpha\to\beta}[\overline{ht_1 \dots t_n}^*, u] &= \lambda \mathsf{a}.\mathbb{S}_\beta[\overline{ht_1 \dots t_n}^* \mathbb{D}_\alpha[\lambda e.\mathsf{a}], u] && \text{(by (4.37c))}\\
&= \lambda \mathsf{a}.\mathbb{S}_\beta[\overline{ht_1 \dots t_n}^* (\mathsf{a}[\mathsf{a} := \mathbb{D}_\alpha[\lambda e.\mathsf{a}]]), u]\\
&= \lambda \mathsf{a}.\mathbb{S}_\beta[\overline{ht_1 \dots t_n}^* (\overline{\mathsf{a}}[\mathsf{a} := \mathbb{D}_\alpha[\lambda e.\mathsf{a}]]), u]\\
& && \text{(since } \mathsf{a} \text{ is a variable)}\\
&= \lambda \mathsf{a}.\mathbb{S}_\beta[\overline{ht_1 \dots t_n}^* \overline{\mathsf{a}}^*, u] && \text{(by 4.41)}\\
&= \lambda \mathsf{a}.\mathbb{S}_\beta[\overline{ht_1 \dots t_n \mathsf{a}}^*, u] && (4.50)
\end{aligned}$$

  By I.H.1 $\mathsf{a} \in \mathsf{Red}_\alpha$ and $ht_1 \dots t_n \mathsf{a} \in \mathsf{Red}_\beta$. Hence, by Remark 4.48, $ht_1 \dots t_n \in \mathsf{Red}_{\alpha\to\beta}$. Moreover, according to I.H.2, the term $ht_1 \dots t_n \mathsf{a}$ is stable. Therefore, by Definition 4.45, (4.50) is equal to (4.51), which is η-equivalent to (4.52):

$$\begin{aligned}
&\lambda \mathsf{a}.ht_1 \dots t_n \mathsf{a} && (4.51)\\
=_\eta\ &ht_1 \dots t_n && (4.52)
\end{aligned}$$

□

Corollary 4.51. *Every non-logical constant is reducible.*

Proof. The corollary is a special case of Item 1 of Lemma 4.50. □

Lemma 4.52. *Every logical constant is reducible modulo logical equivalence.*

Proof.

- $\wedge : o \to o \to o$

  Let p, q, u, v be arbitrary terms, such that $p \in \mathsf{Red}_o$, $q \in \mathsf{Red}_o$, u is of type γ, v is of type o. Let u' be a variable of type γ, such that $u' \notin FV(v)$. Then,

$$\begin{aligned}
\overline{\wedge pq}^* u(\lambda u'.v) &= \overline{\wedge}^* \overline{p}^* \overline{q}^* u(\lambda u'.v)\\
&= \overline{\wedge} \overline{p}^* \overline{q}^* u(\lambda u'.v)\\
&= (\lambda \mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e'.\mathbf{B}e'\phi)) \overline{p}^* \overline{q}^* u(\lambda u'.v) && \text{(by (4.38e))}\\
&\to_\beta^* (\lambda e\phi.\overline{p}^* e(\lambda e'.\overline{q}^* e'\phi)) u(\lambda u'.v)\\
&\to_\beta^* \overline{p}^* u(\lambda e'.\overline{q}^* e'(\lambda u'.v))\\
&= \overline{p}^* u(\lambda e'.\overline{q}^* \wedge v) && \text{(since } q \in \mathsf{Red}_o)\\
&= p \wedge q \wedge v && \text{(since } p \in \mathsf{Red}_o)
\end{aligned}$$

- $\exists : (\iota \to o) \to o$

  Let P, u, v be arbitrary terms such that $\mathrm{P} \in \mathsf{Red}_{\iota\to o}$, u is of type γ, v is of type o. Let u' be a variable of type γ, such that $u' \notin FV(v)$. Then,

$$\begin{aligned}
\overline{\exists\mathrm{P}}^* u(\lambda u'.v) &= \overline{\exists}^*\overline{\mathrm{P}}^* u(\lambda u'.v) \\
&= \overline{\exists\mathrm{P}}^* u(\lambda u'.v) \\
&= (\lambda\mathbf{P}.\lambda e\phi.\exists(\lambda x.\mathbf{P}(\lambda e.x)e\phi))\overline{\mathrm{P}}^* u(\lambda u'.v) && \text{(by (4.38f))} \\
&\to_\beta (\lambda e\phi.\exists(\lambda x.\overline{\mathrm{P}}^*(\lambda e.x)e\phi))u(\lambda u'.v) \\
&\to_\beta (\lambda\phi.\exists(\lambda x.\overline{\mathrm{P}}^*(\lambda e.x)u\phi))(\lambda u'.v) \\
&\to_\beta \exists(\lambda x.\overline{\mathrm{P}}^*(\lambda e.x)u(\lambda u'.v)) \\
&= \exists(\lambda x.\overline{\mathrm{P}}^*\mathbb{D}_\iota[\lambda e.x]u(\lambda u'.v)) \\
&= \exists(\lambda x.\overline{\mathrm{P}x}^* u(\lambda u'.v)) && (4.53)
\end{aligned}$$

  By a the hypothesis $\mathrm{P} \in \mathsf{Red}_{\iota\to o}$. By Lemma 4.50, $x \in \mathsf{Red}_\iota$. Therefore, by Remark 4.48, $\mathrm{P}x \in \mathsf{Red}_o$. Hence, (4.53) is equal to (4.54):

$$\begin{aligned}
&\exists(\lambda x.\mathrm{P}x \wedge v) && (4.54) \\
&= \exists(\lambda x.\mathrm{P}x) \wedge v && (x \notin FV(v))
\end{aligned}$$

- $\neg : o \to o$

  Let p, u, v be arbitrary terms such that $p \in \mathsf{Red}_o$, u is of type γ, v is of type o. Let u' be a variable of type γ, such that $u' \notin FV(v)$. Then,

$$\begin{aligned}
\overline{\neg p}^* u(\lambda u'.v) &= \overline{\neg}^*\overline{p}^* u(\lambda u'.v) \\
&= \overline{\neg p}^* u(\lambda u'.v) \\
&= (\lambda\mathbf{A}.\lambda e\phi.\neg(\mathbf{A}e(\lambda e.\top)) \wedge \phi e)\overline{p}^* u(\lambda u'.v) && \text{(by (4.38g))} \\
&\to_\beta (\lambda e\phi.\neg(\overline{p}^* e(\lambda e.\top)) \wedge \phi e)u(\lambda u'.v) \\
&\to_\beta (\lambda\phi.\neg(\overline{p}^* u(\lambda e.\top)) \wedge \phi u)(\lambda u'.v) \\
&\to_\beta \neg(\overline{p}^* u(\lambda e.\top)) \wedge (\lambda u'.v)u \\
&\to_\beta \neg(\overline{p}^* u(\lambda e.\top)) \wedge v \\
&= \neg(p \wedge \top) \wedge v && \text{(since } p \in \mathsf{Red}_o\text{)} \\
&\equiv \neg p \wedge v
\end{aligned}$$

□

LEMMA 4.53. *Let v and u be two terms of type δ such that $v = u$. If v is reducible, so is u.*

Proof. The proof is by induction on the type δ.

- $\delta = \iota$

  By assumption, $v \in \mathsf{Red}_\iota$, hence for all w, $\overline{v}^* w = v$. Then, since $u = v$, $u = \overline{v}^* w$. Moreover, by Lemma 4.43, $\overline{v}^* = \overline{u}^*$. Therefore, $\overline{u}^* w = u$, i.e. $u \in \mathsf{Red}_\iota$.

- $\delta = o$

  By assumption, $v \in \mathsf{Red}_o$, hence for all w and w' of type γ, p of type o, such that $w' \notin FV(p)$, $\overline{v}^* w(\lambda w'.p) = v \wedge p$. Then, since $u = v$, and, by Lemma 4.43, $\overline{v}^* = \overline{u}^*$, we get $\overline{u}^* w(\lambda w'.p) = u \wedge p$, i.e. $u \in \mathsf{Red}_o$.

- $\delta = \alpha \to \beta$

  We need to prove that for all $w \in \mathsf{Red}_\alpha$, $uw \in \mathsf{Red}_\beta$. Let w' be an arbitrary term in Red_α. Since $v = u$, $vw' = uw'$. Moreover, since $v \in \mathsf{Red}_{\alpha\to\beta}$, $vw' \in \mathsf{Red}_\beta$. Therefore, by I.H., $uw' \in \mathsf{Red}_\beta$, hence $u \in \mathsf{Red}_{\alpha\to\beta}$.

□

Corollary 4.54. *Let t be a term such that for every reducible term u, $t[x := u]$ is reducible. Then, $\lambda x.t$ is reducible.*

Proof. Let t be a term of type β. Let $u \in \mathsf{Red}_\alpha$. Then, $t[x := u] \in \mathsf{Red}_\beta$. From $t[x := u] = (\lambda x.t)u$, it follows that $(\lambda x.t)u \in \mathsf{Red}_\beta$. Therefore, by Definition 4.46, $(\lambda x.t) \in \mathsf{Red}_{\alpha\to\beta}$. □

Lemma 4.55. *Every term t of type δ is reducible.*

Proof. Let free variables of t be among $x_1, \ldots, x_n$ of types $\alpha_1, \ldots, \alpha_n$ correspondingly and let $t_1, \ldots, t_n$ be reducible terms of types $\alpha_1, \ldots, \alpha_n$ correspondingly. We need to prove that $t[x_1 := t_1, \ldots, x_n := t_n]$ is reducible. The proof is by induction on the structure of t.

- If t is a variable x, then $x[x := t_1] =_s t_1$. By assumption, t_1 is reducible, therefore so is $x[x := t_1]$.

- t is a constant k. Follows by Corollary 4.51 and Lemma 4.52.

- If t is an application vu, then

  $$vu[x_1 := t_1, \ldots, x_n := t_n] = v[x_1 := t_1, \ldots, x_n := t_n]u[x_1 := t_1, \ldots, x_n := t_n]$$

  By I.H. $v[x_1 := t_1, \ldots, x_n := t_n]$ and $u[x_1 := t_1, \ldots, x_n := t_n]$ are reducible. Then, so is $v[x_1 := t_1, \ldots, x_n := t_n]u[x_1 := t_1, \ldots, x_n := t_n]$, by Remark 4.48. Hence, $vu[x_1 := t_1, \ldots, x_n := t_n]$ is also reducible by Lemma 4.53.

- If t is an abstraction $\lambda x.v$ of type $(\alpha \to \beta)$, such that $x \notin \{x_1, \ldots, x_n\}$ and $x \notin FV\{t_1, \ldots, t_n\}$. Let u be a term, such that $u \in \mathsf{Red}_\alpha$.

$$\begin{aligned}(\lambda x.v)[x_1 := t_1, \ldots, x_n := t_n]u &= (\lambda x.v[x_1 := t_1, \ldots, x_n := t_n])u \\ &= v[x_1 := t_1, \ldots, x_n := t_n][x := u] \\ &= v[x_1 := t_1, \ldots, x_n := t_n, x := u] \\ &\qquad (\text{since } x \notin FV\{x_1, \ldots, x_n\})\end{aligned}$$

  By I.H., $v[x_1 := t_1, \ldots, x_n := t_n, x := u] \in \mathsf{Red}_\beta$, then so is $(\lambda x.v)[x_1 := t_1, \ldots, x_n := t_n]u$. Then, by Corollary 4.54, $(\lambda x.v)[x_1 := t_1, \ldots, x_n := t_n] \in \mathsf{Red}_{\alpha\to\beta}$.

□

THEOREM 4.56. *For all propositions t, for all terms u of type γ, the following equation holds:*

$$t \equiv \bar{t}u(\lambda e.\top)$$

where e is a variable of type γ.

Proof. By Lemma 4.55, t is reducible, i.e. $t \in \mathsf{Red}_o$. Then, take v equal to $\top$ in Definition 4.46:

$$\bar{t}u(\lambda e.\top) \ = t \wedge \top \equiv t$$

□

DEFINITION 4.57. A dynamic proposition $\mathbf{t}$ is true in a model $\mathcal{M}$, denoted $\mathcal{M} \models_{dyn} \mathbf{t}$, if and only if $\mathcal{M} \models \mathbf{t}u(\lambda e.\top)$ for all u of type γ.

Finally, the conservation theorem can be proved:

THEOREM 4.58. [Conservation] *A proposition t is true in a model $\mathcal{M}$ if and only if its dynamic variant $\bar{t}$ is true in the same model:*

$$\mathcal{M} \models t \textit{ iff } \mathcal{M} \models_{dyn} \bar{t}$$

Proof. If $\mathcal{M} \models t$, then, by Theorem 4.56, $\mathcal{M} \models \bar{t}u(\lambda e.\top)$. Therefore, by Definition 4.57, $\mathcal{M} \models_{dyn} \bar{t}$.

If $\mathcal{M} \models_{dyn} \bar{t}$, then, by Definition 4.57, $\mathcal{M} \models \bar{t}u(\lambda e.\top)$. Therefore, by Theorem 4.56, $\mathcal{M} \models t$. □

Theorem 4.58 guarantees that everything that is valid with respect to static semantics is valid with respect to dynamic semantics, and that the opposite direction also holds.

This chapter, specifically Section 4.3, proposed a compositional dynamic framework GL of natural language semantics in the spirit of Montague Semantics[16]. The proposed framework is promising for handling unconventional phenomena discussed in Chapter 2. It is based on de Groote's [2006] continuation-based approach[17], but significantly elaborates it by considering a more realistic context[18] and by defining a systematic translation from static to dynamic interpretations[19]. Modularity of the approach makes it possible to define interpretations in a compact and intuitive way. Moreover, the conservation theorem guaranteeing that static and dynamic terms hold in the same models is proven.

Unconventional phenomena, such as those discussed in Chapter 2, can be handled by modifying the dynamic interpretations. The next chapter, among other things, demonstrates how this can be accomplished in framework GL for presuppositions triggered by referring expressions. Moreover, the rest of the dissertation is devoted to analysing framework GL and developing it further.

[16] See Section 1.4.
[17] See Chapter 3 and Section 4.1.
[18] See Subsection 2.1.1 and Subsection 4.3.2.
[19] See Subsection 4.3.1.

Chapter 5

From Static to Dynamic Meaning: Interpretation of Lexical Items and Sentences

Montague's theory showing that natural language meaning can be mathematically expressed in a compositional way[1] [1970a; 1970b; 1973], had an important impact on the further development of formal semantics. However, as discussed in Chapter 2, it remained unclear how Montague's ideas could be extended to handle the dynamic behaviour of natural language, such as, for example, cross-sentential pronominal anaphora, donkey anaphora and referring expressions.

A possible extension of Montague's semantics by de Groote [2006], presented in Sections 4.1 and 4.2, uses continuation technique[2] for handling donkey-anaphora. This chapter elaborates de Groote's lexical semantics by using the higher-order continuation-based dynamic logic developed in Section 4.3 to systematically and precisely define lexical meanings. Moreover, as can be seen in the numerous examples of this chapter, it provides a way of representing interpretations in a compact way analogous to familiar static meanings. Furthermore, the fact that the context is treated in a special way significantly raises the expressive power of the framework.

Among unconventional natural language phenomena discussed in Chapter 2, this chapter focuses on presuppositions triggered by referring expressions[3] and takes into account their descriptive content.

Section 5.1 shows how static interpretation of lexical items that do not have extra semantic or pragmatic content and are independent of any context can be dynamized simply by following definitions from Section 4.3.1. Section 5.2 handles interpretations of more complex lexical constituents, specifically referring expressions, which trigger presuppositions of existence of a referent. Section 5.3

[1] See Chapter 1.
[2] See Chapter 3.
[3] Other natural language phenomena are considered in Chapters 6 and 7.

compares standard static interpretations and dynamic interpretations of lexical items discussed in Sections 5.1 and 5.2 and shows that dynamic interpretations are structurally analogous to static interpretations but advantageous with respect to expressing dynamic features of the lexical constituents.

The final Section 5.5 computes and analyses the dynamic meaning of a donkey sentence using the compact dynamic lexical interpretations. This computation not only successfully handles donkey anaphora, but is also as intuitive as the computation using the static interpretations. This can be clearly seen in the final subsection, where the computations of the meaning using static terms and using compact dynamic terms are juxtaposed.

5.1 Dynamic Interpretations Analogous to Static Interpretations

Lexical items analysed in this section do not have any additional linguistic meaning apart from their direct content. Therefore, dynamization of these items is relatively straightforward and can be done according to the definitions in Section 4.3.1.

5.1.1 Nouns

Consider the static interpretation (5.1) of the noun *story*, where **story** is a non-logical constant of type $(\iota \to o)$:

$$[\![story]\!] = \lambda x.\mathbf{story}x \tag{5.1}$$

Example 5.1 computes the normalized dynamic interpretation for the noun according to transformation rules given in Definitions 4.27 and 4.28:

Example 5.1.

$$\begin{aligned}
\overline{[\![story]\!]} &= \overline{\lambda x.\mathbf{story}x} \\
&= \lambda x.\overline{\mathbf{story}x} && \text{(by (4.38d))} \\
&= \lambda x.\overline{\mathbf{story}}\overline{x} && \text{(by (4.38c))} \\
&= \lambda x.\overline{\mathbf{story}}x && \text{(by (4.38b))} \\
&= \lambda x.\mathbb{D}_{\iota\to o}[\lambda e.\mathbf{story}]x && \text{(by (4.38a))} \\
&= \lambda x.(\lambda \mathbf{a}.\mathbb{D}_o[\lambda e.\mathbf{story}\ \mathbb{S}_\iota[\lambda e.\mathbf{a}, e]])x && \text{(by (4.36c))} \\
&= \lambda x.(\lambda \mathbf{a}.\mathbb{D}_o[\lambda e.\mathbf{story}(\mathbf{a}e)])x && \text{(by (4.37a))} \\
&\to_\beta \lambda x.\mathbb{D}_o[\lambda e.\mathbf{story}(xe)] \\
&= \lambda xe\phi.\mathbf{story}(xe) \wedge \phi\ (\mathsf{upd}(\mathbf{story}(xe), e)) && \text{(by (4.36b))}
\end{aligned}$$

□

Hence, the normalized dynamic interpretation of the noun *story* is (5.2). Note that the context in this dynamic interpretation is updated with the proposition $\mathbf{story}(\mathbf{x}e)$:

$$\overline{[\![story]\!]} = \lambda \mathbf{x} e \phi.\mathbf{story}(\mathbf{x}e) \wedge \phi\ (\mathsf{upd}(\mathbf{story}(\mathbf{x}e), e)) \tag{5.2}$$

Since $[\![story]\!]$ is of static type $(\iota \to o)$, $\overline{[\![story]\!]}$ is of dynamic type $(\bar{\iota} \to \bar{o})$. The variable $\mathbf{x}$ in Equation (5.2) is emphasized with bold font as a reminder that it is of dynamic type $\bar{\iota}$.[4]

The term $(\lambda x.\mathbf{story}x)$ is η-equivalent to the term $\mathbf{story}$. Therefore, it is possible to express $\overline{[\![story]\!]}$ even more compactly, as shown in (5.3), which also β-reduces to term (5.2) modulo renaming of bound variables.

$$\overline{[\![story]\!]} = \overline{\mathbf{story}} \tag{5.3}$$

5.1.2 Conventional Verbs

Consider the static type-raised interpretation (5.4) for the transitive verb *love*, where $\mathbf{love}$ is a non-logical constant of type $(\iota \to \iota \to o)$:

$$[\![love]\!] = \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{love}xy)) \tag{5.4}$$

It can be systematically dynamized by following the transformation rules defined in 4.27 and 4.28, as shown in Example 5.2:

Example 5.2.

$$\begin{aligned}
\overline{[\![love]\!]} &= \overline{\lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{love}xy))} \\
&= \lambda \mathrm{YX}.\overline{\mathrm{X}}(\lambda x.\overline{\mathrm{Y}}(\lambda y.\overline{\mathbf{love}xy})) && \text{(by (4.38d))} \\
&= \lambda \mathrm{YX}.\overline{\mathrm{X}}(\lambda x.\overline{\mathrm{Y}}(\lambda y.\overline{\mathbf{love}}\ \overline{x}\ \overline{y})) && \text{(by (4.38c))} \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\overline{\mathbf{love}}xy)) && \text{(by (4.38b))} \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbb{D}_{\iota\to\iota\to o}[\lambda e.\mathbf{love}]xy)) && \text{(by (4.38a))} \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.(\lambda \mathbf{a}.\mathbb{D}_{\iota\to o}[\lambda e.\mathbf{love}\ \mathbb{S}_\iota[\mathbf{a}, e]])xy)) && \text{(by (4.36c))} \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.(\lambda \mathbf{a}.\mathbb{D}_{\iota\to o}[\lambda e.\mathbf{love}(\mathbf{a}e)])xy)) && \text{(by (4.37a))} \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.(\lambda \mathbf{a}\mathbf{a}'.\mathbb{D}_o[\lambda e.\mathbf{love}(\mathbf{a}e)\mathbb{S}_\iota[\mathbf{a}', e]])xy)) && \text{(by (4.36c))} \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.(\lambda \mathbf{a}\mathbf{a}'.\mathbb{D}_o[\lambda e.\mathbf{love}(\mathbf{a}e)(\mathbf{a}'e)])xy)) && \text{(by (4.37a))} \\
&\to^*_\beta \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbb{D}_o[\lambda e.\mathbf{love}(xe)(ye)])) \\
&= \lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.(\lambda e\phi.\mathbf{love}(xe)(ye) \wedge && \text{(by (4.36b))} \\
&\qquad \phi\ (\mathsf{upd}(\mathbf{love}(xe)(ye), e))))
\end{aligned}$$

□

[4] The convention that static variables have regular font and dynamic variables have bold font is, for convenience, pursued throughout the dissertation.

Equation (5.5) repeats the dynamic interpretation above for the verb *love* with the variables in bold, to emphasize that they are of dynamic types:

$$\overline{[\![love]\!]} = \lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.(\lambda e\phi.\mathbf{love}(\mathbf{x}e)(\mathbf{y}e)\wedge \phi\ (\mathsf{upd}(\mathbf{love}(\mathbf{x}e)(\mathbf{y}e), e))))) \tag{5.5}$$

Note that the static version of the verb is of type $(((\iota \to o) \to o) \to ((\iota \to o) \to o) \to o)$ and the dynamic version is of type $(((\overline{\iota} \to \overline{o}) \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o})$. The context is updated with the proposition $\mathbf{love}(\mathbf{x}e)(\mathbf{y}e)$.

The fourth line in the computation of $\overline{[\![love]\!]}$ in Example (5.2), repeated in Equation (5.6), deserves particular attention: it is analogous to the static interpretation $[\![love]\!]$, shown in (5.4), but the constant $\overline{\mathbf{love}}$ is dynamic and variables are of dynamic types:

$$\overline{[\![love]\!]} = \lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy})) \tag{5.6}$$

This shows an important feature of continuation-based dynamic semantics: the types of dynamic terms mirror the types of the corresponding static terms and, structurally, the dynamic terms are analogous to the static ones. Therefore, it changes Montague's lexical interpretations in a minimal and straightforward way. Nevertheless, it can potentially handle dynamic phenomena of natural language because dynamic (sub)terms incorporate the notions of context and continuation.

5.1.3 Indefinite Article

The indefinite article belongs to the syntactic category (n → np). Therefore, the static interpretation of the indefinite article is of type $((\iota \to o) \to ((\iota \to o) \to o))$ and is commonly considered to be the following term:

$$[\![a]\!] = \lambda\mathrm{PQ}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x) \tag{5.7}$$

The dynamic interpretation is a term of type $((\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}))$ systematically obtained as shown below:

$$\begin{aligned}
\overline{[\![a]\!]} &= \overline{\lambda\mathrm{PQ}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)} \\
&= \overline{\lambda\mathrm{PQ}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)} \\
&= \lambda\mathrm{P}\overline{\mathrm{Q}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)} \\
&= \lambda\mathrm{PQ}.\overline{\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)} \\
&= \lambda\mathrm{PQ}.\overline{\exists}\overline{(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)} \\
&= \lambda\mathrm{PQ}.\overline{\exists}(\lambda x.\overline{\mathrm{P}x \wedge \mathrm{Q}x}) \\
&= \lambda\mathrm{PQ}.\overline{\exists}(\lambda x.\overline{\mathrm{P}x}\overline{\wedge}\overline{\mathrm{Q}x}) \\
&= \lambda\mathrm{PQ}.\overline{\exists}(\lambda x.\overline{\mathrm{P}}\overline{x}\overline{\wedge}\overline{\mathrm{Q}}\overline{x}) \\
&= \lambda\mathrm{PQ}.\overline{\exists}(\lambda x.\mathrm{P}x\overline{\wedge}\mathrm{Q}x)
\end{aligned}$$

The resulting dynamic term is analogous to the static interpretation of *a*. The only difference is that all the subterms are now dynamic. To emphasize it, the interpretation is repeated in (5.8) using bold characters:

$$\overline{[\![a]\!]} = \lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx}) \tag{5.8}$$

Interpretation of the indefinite article does not update the context with the contents of its arguments. Nevertheless, these contents are guaranteed to be in the context due to the fact that interpretations of nouns do the context update. However, the indefinite article quantifiers over this context as a result of using continuation technique. Example below illustrates this by computing the meaning of the noun phrase *a Frenchman*:

Example 5.3. [*a Frenchman*] The noun *Frenchman* is interpreted as explained in Subsection 5.1.1:

$$\overline{[\![Frenchman]\!]} = \lambda\mathbf{x}e\phi.\mathbf{Frenchman}(\mathbf{x}e) \wedge \phi(\mathsf{upd}(\mathbf{Frenchman}(\mathbf{x}e), e))$$

$$\begin{aligned}
&\overline{[\![a]\!]}\ \overline{[\![Frenchman]\!]} \\
&= (\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx}))\overline{[\![Frenchman]\!]} \\
&\to_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.\overline{[\![Frenchman]\!]}\mathbf{x} \mathbin{\overline{\wedge}} \mathbf{Qx}) \\
&\to_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.(\mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi))(\overline{[\![Frenchman]\!]}\mathbf{x})(\mathbf{Qx})) \\
&\to^*_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.(\lambda e\phi.(\overline{[\![Frenchman]\!]}\mathbf{x})e(\lambda e.(\mathbf{Qx})e\phi))) \\
&= \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.(\lambda e\phi.((\lambda\mathbf{z}e\phi.\mathbf{Frenchman}(\mathbf{z}e)\wedge \\
&\qquad\qquad \phi\ (\mathsf{upd}(\mathbf{Frenchman}(\mathbf{z}e), e)))\mathbf{x})\ e\ (\lambda e.(\mathbf{Qx})e\phi))) \\
&\to_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.(\lambda e\phi.(\lambda e\phi.\mathbf{Frenchman}(\mathbf{x}e)\wedge \\
&\qquad\qquad \phi\ (\mathsf{upd}(\mathbf{Frenchman}(\mathbf{x}e), e)))\ e\ (\lambda e.(\mathbf{Qx})e\phi))) \\
&\to^*_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.(\lambda e\phi.\mathbf{Frenchman}(\mathbf{x}e) \wedge (\lambda e.(\mathbf{Qx})e\phi)\ (\mathsf{upd}(\mathbf{Frenchman}(\mathbf{x}e), e)))) \\
&\to^*_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.(\lambda e\phi.\mathbf{Frenchman}(\mathbf{x}e) \wedge (\mathbf{Qx})\ (\mathsf{upd}(\mathbf{Frenchman}(\mathbf{x}e), e))\phi))
\end{aligned}$$

Note that the context is updated with the proposition $\mathbf{Frenchman}(\mathbf{x}e)$ and the existential quantifier takes the scope over the whole formula. □

5.1.4 Determiner "every"

Being a determiner, *every* belongs to the syntactic category (n → np). Therefore, the static semantic interpretation of *every* is of type $((\iota \to o) \to ((\iota \to o) \to o))$. The commonly accepted static interpretation is as follows:

$$[\![every]\!] = \lambda\mathrm{PQ}.\forall(\lambda x.\mathrm{P}x \to \mathrm{Q}x) \tag{5.9}$$

The dynamic interpretation of *every* is an analogous term of type $((\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}))$:

$$\begin{aligned}\overline{[\![every]\!]} &= \overline{\lambda \mathrm{PQ}.\forall(\lambda x.\mathrm{P}x \to \mathrm{Q}x)} \\ &= \lambda \mathbf{PQ}.\overline{\forall}(\lambda \mathbf{x}.\mathbf{Px} \Rrightarrow \mathbf{Qx}) \end{aligned} \tag{5.10}$$

Applying abbreviations given in Remark 4.29, term (5.10) can be written as follows:

$$\overline{[\![every]\!]} = \lambda \mathbf{PQ}.\overline{\neg}\overline{\exists}(\lambda \mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \neg\mathbf{Qx}) \tag{5.11}$$

5.1.5 Relative Pronouns

A relative pronoun belongs to the syntactic category $((\mathrm{np} \to \mathrm{s}) \to \mathrm{n} \to \mathrm{n})$. Therefore, the static semantic type of the interpretation of a relative pronoun is $((((\iota \to o) \to o) \to o) \to (\iota \to o) \to (\iota \to o))$. Thus, the term accepts three arguments: R of category $(\mathrm{np} \to \mathrm{s})$ and type $(((\iota \to o) \to o) \to o)$ that corresponds to the relative clause, Q of category n and type $(\iota \to o)$ that corresponds to the antecedent of the relative clause and x of type ι; and returns a proposition of type o. The commonly accepted interpretation of the relative pronoun *who*, for example, is as follows:

$$[\![who]\!] \;=\; \lambda \mathrm{RQ}x.\mathrm{Q}x \wedge \mathrm{R}(\lambda \mathrm{P}.\mathrm{P}x) \tag{5.12}$$

The interpretation in dynamic logic is, once again, analogous to the static interpretation: it is a term of type $((((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}) \to (\overline{\iota} \to \overline{o}) \to (\overline{\iota} \to \overline{o}))$ shown in (5.13). The only difference with the static version is that all the subterms are now dynamic:

$$\begin{aligned}\overline{[\![who]\!]} &= \overline{\lambda \mathrm{RQ}x.\mathrm{Q}x \wedge \mathrm{R}(\lambda \mathrm{P}.\mathrm{P}x)} \\ &= \lambda \mathbf{RQx}.\mathbf{Qx} \mathbin{\overline{\wedge}} \mathbf{R}(\lambda \mathbf{P}.\mathbf{Px}) \end{aligned} \tag{5.13}$$

5.2 Referring Expressions as Exception Triggers

Not all natural language lexical items should be dynamized simply by applying the definitions of Section 4.3.1. Some expressions have non-trivial meaning, as discussed in Chapter 2, and can cause so-called **linguistic side-effects**. Side-effects in natural language can be handled as they are handled in programming languages [Shan, 2005]. This dissertation proposes to use an **exception raising** and **handling** mechanism in the style of ML to account for dynamic linguistic side-effects. As defined in [Milner *et al.*, 1997], **exceptions** are a datatype of error values that are specially treated. An exception is **raised** (created) where the failure occurs and **handled** (analysed) elsewhere - possibly far away. An exception is transmitted by all functions until it reaches the **exception handler**. Exceptions can have arguments and, therefore, can "carry" information that can be used for their handling.

5.2.1 Selection Function and Exceptions

The ability of exceptions to carry information is very useful for formalization of the failure of familiarity presupposition. Consider referring expressions, such as pronouns, proper names, definite descriptions, possessives and genitives. Each of them has a descriptive content, which can be formalized as a property. For example, the descriptive content of the pronoun *he* can be formalized as the property $(\lambda x.\mathbf{human}x \wedge \mathbf{male}x)$. This descriptive content is used by the hearer to retrieve an individual from the context. Analogously, framework GL uses the property to select (using function sel) an individual from the context.

Recall Definition 4.31 of the function sel. It is only partially defined there because the context c may not contain an individual satisfying the property P, i.e. there may be no $\mathtt{a}$ that $c \vdash \mathrm{P}\mathtt{a}$. This corresponds to the failure of familiarity presupposition and exception raising can be used to express this failure. Thus, the definition of sel can be made total in the following way:

Definition 5.4. Let P be a term of type $(\iota \to o)$ and c be a term of type o. Then $\mathsf{sel}\ \mathrm{P}\ c$ is defined as follows:

$$\mathsf{sel}\ \mathrm{P}\ c = \begin{cases} \mathsf{choose}\{\mathtt{a} \mid c \vdash \mathrm{P}\mathtt{a}\} & \text{if } \{c \vdash \mathrm{P}\mathtt{a}\} \neq \emptyset \\ \mathsf{raise}\ \mathtt{AbsentIndividualExc}\ \mathrm{P} & \text{otherwise} \end{cases}$$

According to this definition, when an individual with the required property P is absent in c, the function sel raises exception $\mathtt{AbsentIndividualExc}$ that carries the property P. Operator choose is used in Definition 5.4 to articulate the fact that an appropriate (e.g. more salient) individual having property P has to be chosen in the case there are many candidates.[5]

Example 5.5. Assume **non_human**, **human**, **male**, **donkey** , **farmer** and **unicorn** are constants of type $(\iota \to o)$. Then the following equations hold:

$$\mathsf{sel}(\mathbf{non_human})(\mathbf{donkey}x \wedge \mathbf{non_human}x) = x$$
$$\mathsf{sel}(\lambda z.\mathbf{human}z \wedge \mathbf{male}z)(\mathbf{donkey}x \wedge \mathbf{farmer}y \wedge \mathbf{male}y \wedge \mathbf{human}y) = y$$

$$\mathsf{sel}(\mathbf{human})(\mathbf{donkey}x \wedge \mathbf{non_human}x) = \mathtt{AbsentIndividualExc}\ \mathbf{human}$$

$$\mathsf{sel}(\lambda x.\mathbf{unicorn}x)(\mathbf{donkey}x \wedge \mathbf{farmer}y \wedge \mathbf{male}y \wedge \mathbf{human}y) = \mathtt{AbsentIndividualExc}\ (\lambda x.\mathbf{unicorn}x)$$

□

[5] As was already emphasized earlier, formalization of a realistic context is not the major objective of this work. The goal is to develop a flexible framework capable of acting on any formalization of context. This is specially relevant considering that, as time passes, proposals of even more sophisticated and even more realistic structures of context appear.

The exception (`AbsentIndividualExc` P) is subsequently handled by creating and accommodating a new individual having the property P. Exception raising and handling happens when a sentence is provided some context, which takes place on the level of interpretation of discourse. Therefore, formal details of **accommodation as exception handling** are given in Chapter 6, where a function that updates the meaning of a discourse with the meaning of a new sentence is defined. But before that Subsections from 5.2.2 to 5.2.5 show lexical items causing linguistic side-effects are dynamically interpreted.

While selection function sel is a function of type $((\iota \to o) \to \gamma \to \iota)$, i.e. a function that takes a static property, it is convenient to define a term of type $(\widetilde{(\overline{\iota} \to \overline{o})} \to \gamma \to \iota)$ working on a dynamic property. This term is abbreviated as $\widetilde{\mathsf{sel}}$ and is defined in Equation (5.14a) (Equation (5.14b) shows type assignments):

$$\widetilde{\mathsf{sel}} = \lambda \mathbf{P} e.\ \mathsf{sel}\ (\lambda x.\mathbf{P}(\lambda e.x)e(\lambda e.\top))\ e \tag{5.14a}$$

$$\widetilde{\mathsf{sel}} = \lambda \mathbf{P}^{\overline{\iota}\to\overline{o}} e^{\gamma}.\ \mathsf{sel}\ (\lambda x^{\iota}.\ \overbrace{\mathbf{P}(\overbrace{\lambda e.x}^{\overline{\iota}})}^{(\gamma\to(\gamma\to o)\to o)\to o}\ e\ (\lambda e.\top)\ )\ e \tag{5.14b}$$

Note that $\widetilde{\mathsf{sel}}$ is defined via sel. In the body of $\widetilde{\mathsf{sel}}$ the dynamic property $\mathbf{P}$ is transformed into the static property $(\lambda x.\mathrm{P}(\lambda e.x)e(\lambda e.\top))$ that is given to static sel as an argument.

5.2.2 Proper Names

Consider a static type-raised interpretation (5.15) for a proper name *John*, where $\mathbf{j}$ is a non-logical constant:

$$[\![John]\!] = \lambda \mathrm{P}.\mathrm{P}\mathbf{j} \tag{5.15}$$

Dynamizing it in accordance with the dynamic logic rules of Subsection 4.3.1 leads to (5.16):

$$\begin{aligned}\overline{[\![\mathit{John}]\!]} &= \overline{\lambda \mathrm{P}.\mathrm{P}\mathbf{j}} \\ &= \lambda \mathbf{P}.\overline{\mathrm{P}\mathbf{j}} \\ &= \lambda \mathbf{P}.\overline{\mathrm{P}}\ \bar{\mathbf{j}} \\ &= \lambda \mathbf{P}.\mathbf{P}\ \bar{\mathbf{j}} \\ &= \lambda \mathbf{P}.\mathbf{P}\ \mathbb{D}_{\iota}[\lambda e.\mathbf{j}] \\ &= \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathbf{j}) \end{aligned} \tag{5.16}$$

However, term (5.16) is not the desirable interpretation. One of the reasons comes from the assumption already in (5.15) that the interpretation of *John* uses a non-logical constant **j**. This means that the logical language used for interpreting a natural language should a priory have as many constant as there are individuals and, consequently, the semantic lexicon should have as many entries as there are proper names. This seems infeasible.

Another reason is that (5.16) does not reflect the assumption that proper names (being definite descriptions) have descriptive content that is used to retrieve the individual/referent from a context, i.e. that proper names are presupposition triggers. To express this presuppositional meaning, the dynamic term (5.16) has to be modified.

Recall the λ-term of type $((\iota \to o) \to o)$ standing for *John* in Montague's original semantics, shown in Equation (5.15). As was sketched earlier, instead of having the constant **j**, the dynamic interpretation should attempt to select an individual from the environment based on the descriptive content of the proper name. Assume that for *John* the descriptive content is a property of being named "John".[6] Then, the desirable dynamic interpretation can be formalized as shown in Equation (5.17):

$$\widetilde{[\![\mathit{John}]\!]} = \lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "John"})) \tag{5.17}$$

Note that the type of the dynamic term in Equation (5.17) is the dynamic type of a noun phrase, specifically it is $((\bar{\iota} \to \bar{o}) \to \bar{o})$. Thus, the term, just like its static version, accepts a property and returns a proposition. The only difference is that the property and the proposition are dynamic. Therefore, this new interpretation is capable of causing a linguistic side-effect (if there exists no referent for *John* in the context) when it is a constituent of an interpretation of a complex expression. For example, assume that somebody is telling a new story and the first sentence of this story is (42), containing *John*:

[6] This property of a proper name is used here is as a simple and intuitive approximation that allows to proceed with the development of the framework. The actual descriptive content can be more complex. If it is so, it can be easily integrated into the lexicon.

(42) *The story is about John.*

Assume, without loss of generality, that the context of the interpretation of initial discourse (in which (42) is uttered) is empty. When the interpretation of Sentence (42) is appended to the initial discourse, an exception `AbsentIndividualExc` (named "John") is raised, according to Definition 5.4. This side-effect is identified and the necessary measures (accommodation of the individual) are accomplished. Formal details on how it is implemented are given in Chapter 6.

5.2.3 From Proper Names to Pronouns

As discussed in Subsection 2.1.4, pronominal anaphora is a special case of presupposition having poor descriptive content. If an anaphoric pronoun is uttered in a discourse without an appropriate antecedent, the resulting discourse is not felicitous from a pragmatical point of view. However, this can be fixed with the sentences that come next.

Consider the following dialogue happening in a situation when a lady in love (B) phones her friend (A).

(1) A: Hi! I didn't hear long time from you! How are you?
(2) B: Ah! I am so happy! He seem to be totally into me: he sends flowers, invites me to opera, phones to wish me good night ...
(3) A: Wait ... slow down a bit ... Who is "he"?
(4) B: I have not told you yet? He is a guy I met on Tom's birthday party. His name is Pierre Blaise. He was born in Paris.

In (2) B assumes that A already knows about Pierre. However, A does not. Nevertheless, even though A has never heard about him, sentences as (2) are meaningful for her. It is clear for A that B is talking about a man whom she is probably in love with. Moreover, it is obvious for A, that B presupposes that A already heard about Pierre. A knows that this can lead to problems in understanding what B will continue saying, because B may skip necessary details thinking that A has already heard them. To prevent this and to fill the gap in her knowledge about the man, A explicitly informs B that she is not aware who he is.

This example shows that, from a semantical point of view, the discourse is meaningful at every moment. Therefore, the semantic theory should provide means to express this meaning (even if it is pragmatically incomplete) and to extend the meaning when the lacking descriptive content is eventually provided.

Now imagine a very similar dialogue except that everywhere instead of "he" B and A say "Pierre".

(1) A: Hi! I didn't hear long time from you! How are you?
(2) B: Ah! I am so happy! Pierre seem to be totally into me: he sends flowers,

invites me to opera, phones to wish me good night ...
(3) A: Wait ... slow down a bit ... Who is "Pierre"?
(4) B: I have not told you yet? Pierre is a guy I met on Tom's birthday party. His name is Pierre Blaise. He was born in Paris.

Since A has never heard about Pierre before, the dialogue evolves similarly regardless whether B refers to her admirer as *he* or *Pierre*. In the second case A also interrupts B in order to claim that she lacks some knowledge. There is, however, a small difference. In the first case A knows after (2) only that B talks about a man, while in the second case A learns that B speaks about a person with the name "Pierre". This difference is caused by different descriptive contents of pronouns and proper names.

The dialogues illustrate that pronouns are reminiscent of proper names in that they either refer to a previously introduced individual, or are accommodated in the absence of such an individual. Pronouns, as proper names, belong to the category of noun phrases, and therefore have dynamic type $((\bar{\iota} \to \bar{o}) \to \bar{o})$. This means that the interpretation of pronouns should be analogous to the interpretation of proper names.

The dynamic interpretation of the pronoun *he*, for example, is shown in Equation (5.18), which expresses an attempt to select an individual being a human male. Interpretations of other pronouns are analogous.[7]

$$\widetilde{[\![he]\!]} = \lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)) \tag{5.18}$$

Interpretation (5.18) suits Stalnaker's pragmatic ideas on the meaning of personal pronouns:

> Some sentences may require that a proposition of a certain kind be presupposed without requiring that any particular one of them be presupposed. This is true in general of sentences using demonstratives and personal pronouns. If I say 'he is a linguist', there must be a particular male (the referent of 'he') who is presupposed to exist, but there is no single male whose existence is required by every use of that sentence. In different uses of the sentence, the existence presupposition will be different. [Stalnaker, 1973, p.454]

Consider the following sentence:

(43) *John does not have a car and it is red.*

[7] One might complain that the descriptive content of the pronoun is oversimplified here. Note that this is not an issue investigated in this work. Moreover, and most importantly, the presented framework is flexible to incorporate any descriptive content a pronoun might have: just include the desirable descriptive content in the lexical interpretation of the pronoun.

To make a decision whether this sentence is felicitous or not, one has to first interpret it and framework GL provides means for interpreting the sentence. In GL the pronoun *it* can be interpreted as (5.19) and the meaning of the whole sentence can be computed. Then, the question whether the sentence is felicitous can be resolved depending on context where the sentence is uttered or other constraints.

$$\widetilde{[\![it]\!]} = \lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{non_human}x)) \tag{5.19}$$

5.2.4 Definite Article

The definite article belongs to the syntactic category (n → np), i.e. when given an argument which is a noun, it returns a noun phrase. The resulting definite noun phrase is an anaphoric presupposition triggering phrase. If there is an antecedent in the preceding discourse, it is "bound" to this antecedent. However, if the antecedent is absent, it should be accommodated. However, as in the case of pronouns, the descriptive content is not sufficient for the discourse to be pragmatically felicitous. Consider the dialogue analogous to the dialogues in Section 5.2.3 except that B says "the Frenchman" instead of "Pierre" or "he":

(1) A: Hi! I didn't hear long time from you! How are you?
(2) B: Ah! I am so happy! The Frenchman seem to be totally into me: he sends flowers, invites me to opera, phones to wish me good night ...
(3) A: Wait ... slow down a bit ... Who is "the Frenchman"?
(4) B: I have not told you yet? The Frenchman is a guy I met on Tom's birthday party. His name is Pierre Blaise. He was born in Paris.

Here, as well as in previous versions of the dialogues, B assumes that A already knows about Pierre and that is why she uses the definite phrase "the Frenchman" as a nickname for Pierre. Even though, in fact, it is the first time when A hears about Pierre, the dialogue continues to be meaningful. A just needs to emphasize, as before, that she is lacking some knowledge.

Therefore, a definite noun phrase is behaving like a pronoun (and a proper name) from a semantic point of view: it attempts to select an individual in the current context.[8]

For example, *the Frenchman* can be dynamically interpreted as shown in Equation (5.20). A subterm of this interpretation attempts to select an individual having a property of being a Frenchman from the context e, which is passed as an argument to sel by the abstraction.

$$\widetilde{[\![the]\!]}\overline{[\![Frenchman]\!]} = \lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{Frenchman}x)) \tag{5.20}$$

[8]The treatment of proper names as definite descriptions can be found in [Geurts, 1997; Zeevat, 1999; Beaver, 2001]. A more recent approach that unifies semantics for pronouns, proper names and definite descriptions is [Elbourne, 2005].

However, the dynamic interpretation (5.20) should be computed compositionally, given dynamic interpretations of *the* and *Frenchman*. *Frenchman* is interpreted as a regular dynamic noun:

$$\overline{[\![Frenchman]\!]} = \lambda \mathbf{x} e \phi.\mathbf{Frenchman}(\mathbf{x}e) \wedge \phi(\mathsf{upd}(\mathbf{Frenchman}(\mathbf{x}e), e)) \quad (5.21)$$

The dynamic interpretation of *the* should be a term of type $((\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}))$ (a term that, when given a noun, returns a noun phrase) such that, when the interpretation $\overline{[\![Frenchman]\!]}$ (5.21) is passed to it as an argument, it returns the interpretation $\widetilde{[\![the]\!]}\overline{[\![Frenchman]\!]}$ (5.20). Such a term takes two arguments: a noun $\mathbf{N}$ and a dynamic predicate $\mathbf{P}$. The term should contain as its subterm a selection of an individual having a property given by the noun $\mathbf{N}$. Since $\mathbf{N}$ is a dynamic term, the selection function is also dynamic, particularly, it is the function $\widetilde{\mathsf{sel}}$ defined earlier in Equation (5.14a). Since the function $\widetilde{\mathsf{sel}}$ transforms the dynamic property $\mathbf{P}$ into the static property $(\lambda x.\mathrm{P}(\lambda e.x)e(\lambda e.\top))$ that is given to the static sel as an argument, the discourse update in the interpretation of the noun is guaranteed to be cancelled, as can be seen in Example 5.6.

The resulting dynamic interpretation of the definite article is given by Equation (5.22):

$$\widetilde{[\![the]\!]} = \lambda \mathbf{NP}.\mathbf{P}(\widetilde{\mathsf{sel}}\mathbf{N}) \quad (5.22)$$

Now, after applying interpretation $\widetilde{[\![the]\!]}$ to interpretation $\overline{[\![Frenchman]\!]}$ and β-reducing, the interpretation (5.24) of the definite description *the Frenchman* is compositionally obtained:

Example 5.6. [*the Frenchman*]

$$\begin{aligned}
&\widetilde{[\![the]\!]}\ \overline{[\![Frenchman]\!]} \\
&= (\lambda \mathbf{NP}.\mathbf{P}(\widetilde{\mathsf{sel}}\mathbf{N}))\overline{[\![Frenchman]\!]} && \text{(by (5.22))} \\
&\to^*_\beta \lambda \mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}\overline{[\![Frenchman]\!]}) && (5.23) \\
&= \lambda \mathbf{P}.\mathbf{P}((\lambda \mathbf{P} e.\mathsf{sel}(\lambda x.\mathbf{P}(\lambda e.x)e(\lambda e.\top))e)\overline{[\![Frenchman]\!]}) && \text{(by (5.14a))} \\
&\to_\beta \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\overline{[\![Frenchman]\!]}(\lambda e.x)e(\lambda e.\top))e) \\
&= \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.(\lambda \mathbf{x} e\phi.\mathbf{Frenchman}(\mathbf{x}e) \wedge && \text{(by (5.21))} \\
&\qquad\qquad \phi(\mathsf{upd}(\mathbf{Frenchman}(\mathbf{x}e), e)))(\lambda e.x)e(\lambda e.\top))e) \\
&\to_\beta \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.(\lambda e\phi.\mathbf{Frenchman}((\lambda e.x)e) \wedge \\
&\qquad\qquad \phi(\mathsf{upd}(\mathbf{Frenchman}((\lambda e.x)e), e)))e(\lambda e.\top))e) \\
&\to^*_\beta \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.(\lambda e\phi.\mathbf{Frenchman}x \wedge \phi(\mathsf{upd}(\mathbf{Frenchman}x, e)))e(\lambda e.\top))e) \\
&\to^*_\beta \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\mathbf{Frenchman}x \wedge (\lambda e.\top)(\mathsf{upd}(\mathbf{Frenchman}x, e)))e) \\
&\to_\beta \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\mathbf{Frenchman}x \wedge \top)e) \\
&\equiv \lambda \mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\mathbf{Frenchman}x)e) && (5.24)
\end{aligned}$$

The resulting term (5.24) is η-equivalent to the desired interpretation (5.20). □

5.2.5 Genitives

Consider Sentence (44), where a possessive noun phrase *John's car* triggers a presupposition that there is a car owned by John:

(44) *John's car is red.*

The desired interpretation of *John's car* is shown in (5.25). This interpretation requires a search in the context for an individual having the property of being a car possessed by an individual named "John". Note that the interpretation of the presupposition triggered by *John* (encoded via the inner selection function taking (named "John") as the first argument) is a subterm of the interpretation of another presupposition related to *car* (encoded via the outer selection function).

$$\begin{aligned}&\widetilde{[\!['s]\!]}\ \widetilde{[\![John]\!]}\ \overline{[\![car]\!]}\\ &=\ \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\mathbf{car}x \wedge \mathbf{poss}(\mathsf{sel}(\text{named "John"})e')x)e)\end{aligned} \tag{5.25}$$

where e' is the context e possibly updated with some additional information. Interpretation (5.25) has to be computed compositionally in terms of its constituents. This requires the formalization of the *'s* clitic. Term (5.26a) shows a possible interpretation (type assignments are shown in (5.26b)): it takes a noun phrase standing for a possessor and a noun standing for an object being possessed, and returns a noun phrase in form of (5.25):

$$\widetilde{[\!['s]\!]} = \lambda\mathbf{Y}\mathbf{X}.\lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\mathbf{X}\mathbf{x})\overline{\wedge}\mathbf{Y}(\overline{\mathbf{poss}}\mathbf{x})))) \tag{5.26a}$$

$$\widetilde{[\!['s]\!]} = \lambda\mathbf{Y}^{(\bar{\iota}\to\bar{o})\to\bar{o}}\mathbf{X}^{\bar{\iota}\to\bar{o}}.\lambda\mathbf{P}^{\bar{\iota}\to\bar{o}}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}^{\bar{\iota}}.((\overbrace{\mathbf{X}\mathbf{x}}^{\bar{o}})\overline{\wedge}\mathbf{Y}(\overbrace{\overline{\mathbf{poss}}\mathbf{x}}^{\bar{\iota}\to\bar{o}})))) \tag{5.26b}$$

The term $\overline{[\![poss]\!]}$ in (5.26a) is a usual dynamic two-arguments dynamic constant, its λ-term is shown in (5.27):

$$\overline{\mathbf{poss}} = \lambda\mathbf{x}\mathbf{y}.\lambda e\phi.\mathbf{poss}(\mathbf{x}e)(\mathbf{y}e) \wedge \phi(\mathsf{upd}(\mathbf{poss}(\mathbf{x}e)(\mathbf{y}e), e)) \tag{5.27}$$

After applying the term $\widetilde{[\!['s]\!]}$ (5.26a) to the term $\widetilde{[\![John]\!]}$ (5.17) and the term $\overline{[\![car]\!]}$ (5.28), which is just a dynamic unary predicate, and β-reducing, the desired term (5.25) ((5.30) below) is obtained.

$$\overline{[\![car]\!]} = \lambda\mathbf{x}.\lambda e\phi.\mathbf{car}(\mathbf{x}e) \wedge \phi(\mathsf{upd}(\mathbf{car}(\mathbf{x}e), e)) \tag{5.28}$$

The next example illustrates this in detail.

EXAMPLE 5.7. [*John's car*]

$$\begin{aligned}
&\widetilde{[\!['s]\!]}\ \widetilde{[\![John]\!]}\ \overline{[\![car]\!]}\\
&= (\lambda\mathbf{YX}.\lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\mathbf{Xx})\overline{\wedge}\mathbf{Y}(\overline{\mathbf{poss}}\mathbf{x})))))\widetilde{[\![John]\!]}\ \overline{[\![car]\!]}\\
&\to^*_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\overline{[\![car]\!]}\mathbf{x})\overline{\wedge}\widetilde{[\![John]\!]}(\overline{\mathbf{poss}}\mathbf{x})))) \qquad (5.29)
\end{aligned}$$

Note that it is already possible to see in (5.29) that the interpretation of *John's car* attempts to select an individual having property of being a car and being possessed by an individual named "John". Nevertheless, the term can be normalized by unfolding the definitions of $\widetilde{\mathsf{sel}}$ and the dynamic conjunction $\overline{\wedge}$:

$$\begin{aligned}
&= \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\overline{[\![car]\!]}\mathbf{x})\overline{\wedge}(\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "John"}))(\overline{\mathbf{poss}}\mathbf{x})))) \quad \text{(by (5.17))}\\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\overline{[\![car]\!]}\mathbf{x})\overline{\wedge}((\overline{\mathbf{poss}}\mathbf{x})(\mathsf{sel}(\text{named "John"}))))))\\
&= \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\overline{[\![car]\!]}\mathbf{x})\overline{\wedge}(((\lambda\mathbf{zy}.\lambda e\phi.\mathbf{poss}(\mathbf{z}e)(\mathbf{y}e)\wedge \quad \text{(by (5.27))}\\
&\qquad \phi(\mathsf{upd}(\mathbf{poss}(\mathbf{z}e)(\mathbf{y}e),e))\phi e)\mathbf{x})(\mathsf{sel}(\text{named "John"}))))))\\
&= \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\overline{[\![car]\!]}\mathbf{x})\overline{\wedge}((\lambda\mathbf{y}.\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{y}e)\wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{y}e),e))\phi e)\mathbf{x}))))\\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\overline{[\![car]\!]}\mathbf{x})\overline{\wedge}(\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e))\phi e))))\\
&= \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.(((\lambda\mathbf{x}.\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\phi(\mathsf{upd}(\mathbf{car}(\mathbf{x}e),e)))\mathbf{x})\overline{\wedge} \quad \text{(by (5.28))}\\
&\qquad (\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge\\
&\qquad\quad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e))))))\\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\phi(\mathsf{upd}(\mathbf{car}(\mathbf{x}e),e)))\overline{\wedge}\\
&\qquad (\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge\\
&\qquad\quad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e))))))\\
&= \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\lambda\mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi)) \quad \text{(by (4.38e))}\\
&\qquad (\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\phi(\mathsf{upd}(\mathbf{car}(\mathbf{x}e)(\mathbf{y}e),e)))\\
&\qquad (\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge\\
&\qquad\quad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e))))))\\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\lambda\mathbf{B}.\lambda e\phi.(\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\phi(\mathsf{upd}(\mathbf{car}(\mathbf{x}e)(\mathbf{y}e),e)))e(\lambda e.\mathbf{B}e\phi))\\
&\qquad (\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge\\
&\qquad\quad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e))))))\\
&\to^*_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\lambda\mathbf{B}.\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\mathbf{B}(\mathsf{upd}(\mathbf{car}(\mathbf{x}e)(\mathbf{y}e),e))\phi)\\
&\qquad (\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge\\
&\qquad\quad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e))))))
\end{aligned}$$

$$\begin{aligned}
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.(\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge \\
&\qquad\qquad (\lambda e\phi.\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e)\wedge \\
&\qquad\qquad\qquad \phi(\mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e)(\mathbf{x}e),e)) \\
&\qquad\qquad (\mathsf{upd}(\mathbf{car}(\mathbf{x}e)(\mathbf{y}e),e)))\phi))) \\
&\to^*_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.(\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)(\mathbf{x}e^*)\wedge\phi e^{**})))
\end{aligned}$$

where $e^* = \mathsf{upd}(\mathbf{car}(\mathbf{x}e)(\mathbf{y}e),e)$
and $e^{**} = \mathsf{upd}(\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)(\mathbf{x}e^*),e^*)$

$$\begin{aligned}
&= \lambda\mathbf{P}.\mathbf{P}((\lambda\mathbf{P}e.\mathsf{sel}(\lambda x.\mathbf{P}(\lambda e.x)e(\lambda e.\top))e) && \text{(by (5.14a))} \\
&\qquad (\lambda\mathbf{x}.(\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)(\mathbf{x}e^*)\wedge\phi e^{**}))) \\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.(\lambda\mathbf{x}.(\lambda e\phi.\mathbf{car}(\mathbf{x}e)\wedge \\
&\qquad\qquad \mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)(\mathbf{x}e^*)\wedge\phi e^{**}))(\lambda e.x)e(\lambda e.\top))e) \\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.(\lambda e\phi.\mathbf{car}((\lambda e.x)e)\wedge \\
&\qquad\qquad \mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)((\lambda e.x)e^*)\wedge\phi e^{**})e(\lambda e.\top))e) \\
&\to^*_\beta \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.(\lambda e\phi.\mathbf{car}x\wedge\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)x\wedge\phi e^{**})e(\lambda e.\top))e) \\
&\to^*_\beta \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\mathbf{car}x\wedge\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)x\wedge(\lambda e.\top)e^{**})e) \\
&\to_\beta \lambda\mathbf{P}.\mathbf{P}((\lambda e.\mathsf{sel}(\lambda x.\mathbf{car}x\wedge\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)x\wedge\top)e)) \\
&\equiv \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}(\lambda x.\mathbf{car}x\wedge\mathbf{poss}(\mathsf{sel}(\text{named "John"})e^*)x)e) && (5.30)
\end{aligned}$$

□

Possessive pronouns can also be interpreted using $\widetilde{[\!['s]\!]}$. For example, dynamic interpretation of *its* can be obtained by applying $\widetilde{[\!['s]\!]}$ to the dynamic interpretation $\widetilde{[\![it]\!]}$.

5.3 Comparison of Dynamic and Static Lexical Interpretations

Table 5.1 summarizes dynamic lexical interpretations discussed in this section. For the sake of comparison, Table 5.2 shows static interpretations for the same lexical items. For every lexical item, its dynamic type is analogous to its static type with the only difference that each atomic type is dynamized. The dynamization of static interpretations of the lexical items that do not cause side-effects, such as *story*, *loves*, *a*, *every* and *who*, follow the rules defined in Chapter 4 that ensure that the dynamic terms are structurally analogous to the static ones. Dynamic interpretations of the lexical items that can cause linguistic side-effects, such as *John*, *he*, *the* and genitive are modified to account for these side-effects. The modifications comply with the types of the lexical items.

Lexical item	Syntactic category	Dynamic type	Dynamic interpretation in GL
story	n	$\overline{\iota} \to \overline{o}$	$\overline{\mathbf{story}}$
loves	np $\to$ np $\to$ s	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{lovexy}}))$
a	n $\to$ np	$(\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o})$	$\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx})$
every	n $\to$ np	$(\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o})$	$\lambda\mathbf{PQ}.\overline{\forall}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\Rightarrow}} \mathbf{Qx})$
who	(np $\to$ s) $\to$ n $\to$ n	$(((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}) \to (\overline{\iota} \to \overline{o}) \to (\overline{\iota} \to \overline{o})$	$\lambda\mathbf{RQx}.\mathbf{Qx} \mathbin{\overline{\wedge}} \mathbf{R}(\lambda\mathbf{P}.\mathbf{Px})$
John	np	$(\overline{\iota} \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named “John”}))$
he	np	$(\overline{\iota} \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))$
the	n $\to$ np	$(\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o})$	$\lambda\mathbf{NP}.\mathbf{P}(\widetilde{\mathsf{sel}}\mathbf{N})$
's	np $\to$ n $\to$ np	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to (\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o})$	$\lambda\mathbf{YX}.\lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\mathbf{Xx})\overline{\wedge}\mathbf{Y}(\overline{[\![poss]\!]}\mathbf{x}))))$

Table 5.1: Dynamic lexical interpretations in framework GL.

Lexical item	Syntactic category	Static type	Static interpretation
story	n	$\iota \to o$	$\mathbf{story}$
loves	np $\to$ np $\to$ s	$((\iota \to o) \to o) \to ((\iota \to o) \to o) \to o$	λYX.X(λx.Y($\lambda y.\mathbf{love}xy$))
a	n $\to$ np	$(\iota \to o) \to ((\iota \to o) \to o)$	λPQ.$\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x)$
every	n $\to$ np	$(\iota \to o) \to ((\iota \to o) \to o)$	λPQ.$\forall(\lambda x.\mathrm{P}x \to \mathrm{Q}x)$
who	(np $\to$ s) $\to$ n $\to$ n	$(((\iota \to o) \to o) \to o) \to (\iota \to o) \to (\iota \to o)$	λRQx.Q$x \wedge$ R(λP.Px)
John	np	$(\iota \to o) \to o$	λP.P**j**, where **j** is a constant
he	np	$(\iota \to o) \to o$	λP.P?
the	n $\to$ np	$(\iota \to o) \to ((\iota \to o) \to o)$	λNP.P?
's	np $\to$ n $\to$ np	$((\iota \to o) \to o) \to (\iota \to o) \to ((\iota \to o) \to o)$	λYX.λP.P?

Table 5.2: Static lexical interpretations.

Lexical item	Syntactic category	Dynamic interpretation in GL
story	n	$\overline{\mathbf{story}}$
John	np	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "John"}))$
Mary	np	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "Mary"}))$
he	np	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))$
her	np	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x))$
the	n $\rightarrow$ np	$\lambda\mathbf{NP}.\mathbf{P}(\widetilde{\mathsf{sel}}\mathbf{N})$
loves	np $\rightarrow$ np $\rightarrow$ s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy}))$
smiles_at	np $\rightarrow$ np $\rightarrow$ s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{smilex}}\mathbf{y}))$
is_about	np $\rightarrow$ np $\rightarrow$ s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{about}}\mathbf{xy}))$

Table 5.3: Dynamic lexical interpretation of a tiny fragment of English in framework GL.

5.4 Sentence Meaning

This subsection demonstrates the compositional computation of dynamic meanings of individual sentences composing Discourse (45):

(45) *The story is about John. John loves Mary. He smiles at her.*

Table 5.3 presents the dynamic meanings of lexical items that are the basic constituents of the sentences. The interpretations of these items are given in accordance with discussion in Sections 5.1 and 5.2. The function-argument relation in functional application of meanings of basic constituents is based upon the syntactic trees of the sentences.

Example 5.8. [*The story is about John*] This example compositionally computes the dynamic meaning for Sentence (46):

(46) *The story is about John.*

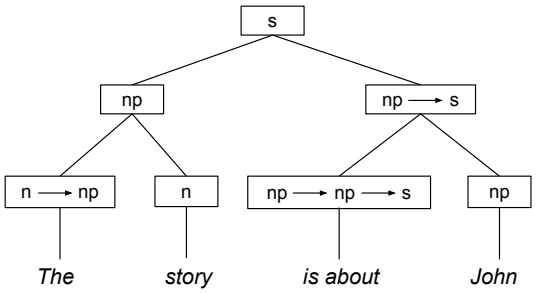


Figure 5.1: Syntactic parse tree of sentence *The story is about John.*

According to the syntactic parse tree of the sentence, presented in Figure 5.1, the semantic interpretation of the sentence can be computed by β-reducing the λ-term (5.31):

$$(\overline{[\![is_about]\!]}\ \widetilde{[\![John]\!]})\ (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]}) \tag{5.31}$$

Application $\widetilde{[\![the]\!]}\ \overline{[\![story]\!]}$ β-reduces as follows:

$$\begin{aligned}\widetilde{[\![the]\!]}\ \overline{[\![story]\!]} &= (\lambda\mathbf{NP}.\mathbf{P}(\widetilde{\mathsf{sel}}\mathbf{N}))\overline{[\![story]\!]}\\ &\rightarrow_\beta \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}\overline{[\![story]\!]})\\ &= \lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\overline{\mathbf{story}}))\end{aligned} \tag{5.32}$$

Note that the term in (5.32) interpreting the noun phrase *the story* is very compact and rather intuitive: it selects the story from the context to be provided (recall that the context is one of the arguments of the function $\widetilde{\mathsf{sel}}$, see Equation (5.14a)).

The meaning $\mathbf{S}_1$ of the sentence itself can now be computed:

$$\begin{aligned}\mathbf{S}_1 &= (\overline{[\![is_about]\!]}\ \widetilde{[\![John]\!]})\ (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})\\ &= (\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{about}}\mathbf{xy})))\widetilde{[\![John]\!]}\ (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})\\ &\rightarrow_\beta (\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\widetilde{[\![John]\!]}(\lambda\mathbf{y}.\overline{\mathbf{about}}\mathbf{xy})))(\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})\\ &\rightarrow_\beta (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})(\lambda\mathbf{x}.\widetilde{[\![John]\!]}(\lambda\mathbf{y}.\overline{\mathbf{about}}\mathbf{xy}))\\ &= (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})(\lambda\mathbf{x}.(\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "John"})))(\lambda\mathbf{y}.\overline{\mathbf{about}}\mathbf{xy}))\\ &\rightarrow_\beta (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})(\lambda\mathbf{x}.(\lambda\mathbf{y}.\overline{\mathbf{about}}\mathbf{xy})(\mathsf{sel}(\text{named "John"})))\\ &\rightarrow_\beta (\widetilde{[\![the]\!]}\ \overline{[\![story]\!]})(\lambda\mathbf{x}.\overline{\mathbf{about}}\ \mathbf{x}(\mathsf{sel}(\text{named "John"})))\\ &= (\lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\overline{\mathbf{story}})))(\lambda\mathbf{x}.\overline{\mathbf{about}}\ \mathbf{x}(\mathsf{sel}(\text{named "John"})))\\ &\rightarrow_\beta (\lambda\mathbf{x}.\overline{\mathbf{about}}\ \mathbf{x}(\mathsf{sel}(\text{named "John"})))(\widetilde{\mathsf{sel}}(\overline{\mathbf{story}}))\\ &\rightarrow_\beta \overline{\mathbf{about}}\ (\widetilde{\mathsf{sel}}(\overline{\mathbf{story}}))(\mathsf{sel}(\text{named "John"})))\end{aligned} \tag{5.33}$$

The term in (5.33) interprets compactly the meaning of the sentence. It includes the presuppositional meanings of the definite noun phrase and pronoun. The interpretation contains dynamic subterms $\overline{\mathbf{about}}$, $\widetilde{\mathsf{sel}}$ and $\overline{\mathbf{story}}$. These subterms ensure that the interpretation of the sentence itself is dynamic. This can be more easily seen after normalizing (5.33) resulting in (5.34):

$$\begin{aligned}&\overline{\mathbf{about}}\ (\widetilde{\mathsf{sel}}(\overline{\mathbf{story}}))(\mathsf{sel}(\text{named "John"}))\\ =\ &\overline{\mathbf{about}}\ (\widetilde{\mathsf{sel}}(\lambda\mathbf{x}e\phi.\mathbf{story}(\mathbf{x}e)\wedge\phi(\mathsf{upd}(\mathbf{story}(\mathbf{x}e),e))))(\mathsf{sel}(\text{named "John"}))\end{aligned} \tag{5.34}$$

$$
\begin{aligned}
&= \overline{\textbf{about}}\ (\lambda \mathbf{P} e'.\mathsf{sel}(\lambda x.\mathbf{P}(\lambda e''.x)e'(\lambda e'''.\top))e') \\
&\qquad (\lambda \mathbf{x} e \phi.\mathbf{story}(\mathbf{x}e) \wedge \phi(\mathsf{upd}(\mathbf{story}(\mathbf{x}e), e)))(\mathsf{sel}(\text{named "John"})) \\
&\to_\beta \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.(\lambda \mathbf{x} e \phi.\mathbf{story}(\mathbf{x}e) \wedge \\
&\qquad \phi(\mathsf{upd}(\mathbf{story}(\mathbf{x}e), e)))(\lambda e''.x)e'(\lambda e'''.\top))e')(\mathsf{sel}(\text{named "John"})) \\
&\to_\beta \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.(\lambda e \phi.\mathbf{story}((\lambda e''.x)e) \wedge \\
&\qquad \phi(\mathsf{upd}(\mathbf{story}((\lambda e''.x)e), e)))e'(\lambda e'''.\top))e')(\mathsf{sel}(\text{named "John"})) \\
&\to_\beta^* \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.(\lambda e \phi.\mathbf{story}x \wedge \phi(\mathsf{upd}(\mathbf{story}(x, e)))e'(\lambda e'''.\top))e') \\
&\qquad (\mathsf{sel}(\text{named "John"})) \\
&\to_\beta \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.(\lambda \phi.\mathbf{story}x \wedge \phi(\mathsf{upd}(\mathbf{story}(x, e'))))(\lambda e'''.\top))e') \\
&\qquad (\mathsf{sel}(\text{named "John"})) \\
&\to_\beta \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.(\mathbf{story}x \wedge (\lambda e'''.\top)(\mathsf{upd}(\mathbf{story}(x, e'))))e') \\
&\qquad (\mathsf{sel}(\text{named "John"})) \\
&\to_\beta \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.(\mathbf{story}x \wedge \top))e')(\mathsf{sel}(\text{named "John"})) \\
&\equiv \overline{\textbf{about}}\ (\lambda e'.\mathsf{sel}(\lambda x.\mathbf{story}x)e')(\mathsf{sel}(\text{named "John"})) \\
&= (\lambda \mathbf{x}\mathbf{y}.\lambda e \phi.\mathbf{about}(\mathbf{x}e)(\mathbf{y}e) \wedge \phi(\mathsf{upd}(\mathbf{about}(\mathbf{x}e)(\mathbf{y}e), e))) \\
&\quad (\lambda e'.\mathsf{sel}(\lambda x.(\mathbf{story}x)e')(\mathsf{sel}(\text{named "John"})) \\
&\to_\beta (\lambda \mathbf{y}.\lambda e \phi.\mathbf{about}((\lambda e'.\mathsf{sel}(\lambda x.\mathbf{story}x)e')e)(\mathbf{y}e) \wedge \\
&\qquad \phi(\mathsf{upd}(\mathbf{about}((\lambda e'.\mathsf{sel}(\lambda x.\mathbf{story}x)e')e)(\mathbf{y}e), e))) \\
&\qquad (\mathsf{sel}(\text{named "John"})) \\
&\to_\beta^* (\lambda \mathbf{y}.\lambda e \phi.\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathbf{y}e) \wedge \\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathbf{y}e), e)))(\mathsf{sel}(\text{named "John"})) \\
&\to_\beta \lambda e \phi.\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e) \wedge \qquad (5.34) \\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e), e))
\end{aligned}
$$

Consider the subterm

$$(\phi\ (\mathsf{upd}(\mathbf{about}\ (\mathsf{sel}(\lambda x.\mathbf{story}x)\ e)\ (\mathsf{sel}(\text{named "John"})\ e), e)))$$

in (5.34). It updates the context e with the proposition that the story is about john and this updated context is given as an argument to the continuation ϕ. Thus, since continuation is "the future of the computation", this updated context is available for the computation of the meaning of a discourse in which this sentence appear. Note, moreover, that the interpretation (5.34) is an abstraction over the context e and the continuation ϕ. This means that the context variable e is to be substituted (due to β-reduction) by the context of the discourse in which the sentence is uttered (when the meaning of the discourse containing this sentence is computed). □

EXAMPLE 5.9. [*John loves Mary*] This example shows the compositional computation of the dynamic meaning $\mathbf{S}_2$ of Sentence (47). The dynamic interpretations for the lexical items of the sentence can be found in Table 5.3.

(47) *John loves Mary*

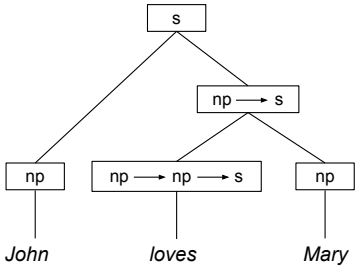


Figure 5.2: Syntactic parse tree of sentence *John loves Mary*.

Recall the parse tree from Figure 1.6, for convenience repeated in Figure 5.2. According to this tree, the meaning of the sentence can be computed by β-reducing the term (5.35):

$$
\begin{aligned}
\mathbf{S}_2 &= (\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]})\ \widetilde{[\![John]\!]} && (5.35)\\
&= (\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy})))\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}\\
&\rightarrow_\beta (\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\widetilde{[\![Mary]\!]}(\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy})))\widetilde{[\![John]\!]}\\
&\rightarrow_\beta \widetilde{[\![John]\!]}(\lambda\mathbf{x}.\widetilde{[\![Mary]\!]}(\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy}))\\
&= \widetilde{[\![John]\!]}(\lambda\mathbf{x}.(\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "Mary"})))(\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy}))\\
&\rightarrow_\beta \widetilde{[\![John]\!]}(\lambda\mathbf{x}.((\lambda\mathbf{y}.\overline{\mathbf{love}}\mathbf{xy})(\mathsf{sel}(\text{named "Mary"}))))\\
&\rightarrow_\beta \widetilde{[\![John]\!]}(\lambda\mathbf{x}.\overline{\mathbf{love}}\ \mathbf{x}\ (\mathsf{sel}(\text{named "Mary"})))\\
&= (\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named "John"})))(\lambda\mathbf{x}.\overline{\mathbf{love}}\ \mathbf{x}\ (\mathsf{sel}(\text{named "Mary"})))\\
&\rightarrow_\beta (\lambda\mathbf{x}.\overline{\mathbf{love}}\ \mathbf{x}\ (\mathsf{sel}(\text{named "Mary"})))(\mathsf{sel}(\text{named "John"}))\\
&\rightarrow_\beta \overline{\mathbf{love}}(\mathsf{sel}(\text{named "John"}))(\mathsf{sel}(\text{named "Mary"})) && (5.36)
\end{aligned}
$$

The term (5.36) is the compact and intuitive dynamic interpretation of the sentence. It can be easily reduced to the normal form (5.37):

$$
\begin{aligned}
&\overline{\mathbf{love}}(\mathsf{sel}(\text{named "John"}))(\mathsf{sel}(\text{named "Mary"}))\\
&= (\lambda\mathbf{xy}.\lambda e\phi.\mathbf{love}(\mathbf{x}e)(\mathbf{y}e) \wedge \phi(\mathsf{upd}(\mathbf{love}(\mathbf{x}e)(\mathbf{y}e), e)))\\
&\quad (\mathsf{sel}(\text{named "John"}))(\mathsf{sel}(\text{named "Mary"}))\\
&\rightarrow^*_\beta \lambda e\phi.\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e)\wedge\\
&\qquad\qquad \phi(\mathsf{upd}(\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e), e)) \qquad (5.37)
\end{aligned}
$$

□

Compare interpretation (5.37) with de Groote's [2006] interpretation (4.9) for the same sentence, repeated below in (5.38):

$$\lambda e\phi.\mathbf{lovejm} \wedge \phi(\mathbf{j} :: \mathbf{m} :: e) \tag{5.38}$$

Interpretation (5.37) is advantageous, because it does not use constants for expressing *John* and *Mary* (unlike (5.38)). It expects to select the required referents from the context e, which is to be provided as an argument to the term. It is more realistic and expresses the presuppositional content of the sentence. Note also that the context in (5.37) (which is an argument of the interpretation) is updated with the proposition that John loves Mary, while the context in (5.38), being simpler, is updated only with the constants standing for *John* and *Mary*.[9]

Example 5.10. [*He smiles at her*] This example computes the dynamic meaning $\mathbf{S}_3$ of Sentence (48). The dynamic interpretations for the lexical items of the sentence can be found in Table 5.3.

(48) *He smiles at her.*

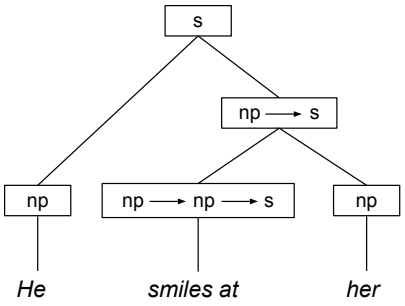


Figure 5.3: Syntactic parse tree of sentence *He smiles at her.*

Recall the parse tree for the sentence shown in Figure 4.2 and repeated in Figure 5.3. According to this parse tree, the meaning of the sentence can be

[9]Term (5.37) appears to be more complex than de Groote's [2006] (5.38). However, as can be easily seen (5.37) and (5.38) are structurally analogous. The apparent complexity of (5.37) is caused by the fact that it accounts for presuppositional content (while (5.38) is not). Moreover, one can simply dwell on term (5.34) (without proceeding with normalization). It is β-equivalent to (5.37), but more compact and intuitive.

computed by β-reducing the term (5.39):

$$
\begin{aligned}
\mathbf{S}_3 &= \overline{[\![smiles_at]\!]}\ \widetilde{[\![her]\!]}\ \widetilde{[\![he]\!]} && (5.39)\\
&= (\lambda \mathbf{YX}.\mathbf{X}(\lambda \mathbf{x}.\mathbf{Y}(\lambda \mathbf{y}.\overline{\mathbf{smile}}\mathbf{xy})))\widetilde{[\![her]\!]}\ \widetilde{[\![he]\!]}\\
&\rightarrow_\beta (\lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\widetilde{[\![her]\!]}(\lambda \mathbf{y}.\overline{\mathbf{smile}}\mathbf{xy})))\widetilde{[\![he]\!]}\\
&\rightarrow_\beta \widetilde{[\![he]\!]}(\lambda \mathbf{x}.\widetilde{[\![her]\!]}(\lambda \mathbf{y}.\overline{\mathbf{smile}}\mathbf{xy}))\\
&= \widetilde{[\![he]\!]}(\lambda \mathbf{x}.(\lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)))(\lambda \mathbf{y}.\overline{\mathbf{smile}}\mathbf{xy}))\\
&\rightarrow_\beta \widetilde{[\![he]\!]}(\lambda \mathbf{x}.((\lambda \mathbf{y}.\overline{\mathbf{smile}}\mathbf{xy})(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)))\\
&\rightarrow_\beta \widetilde{[\![he]\!]}(\lambda \mathbf{x}.\overline{\mathbf{smile}}\ \mathbf{x}\ (\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)))\\
&= (\lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)))\\
&\quad (\lambda \mathbf{x}.\overline{\mathbf{smile}}\ \mathbf{x}\ (\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)))\\
&\rightarrow_\beta (\lambda \mathbf{x}.\overline{\mathbf{smile}}\ \mathbf{x}\ (\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)))(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))\\
&\rightarrow_\beta \overline{\mathbf{smile}}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)) && (5.40)
\end{aligned}
$$

The term (5.40) is the dynamic compact interpretation of the sentence. It can be reduced to the normal form (5.41):

$$
\begin{aligned}
&\overline{\mathbf{smile}}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x))\\
=\ &(\lambda \mathbf{xy}.\lambda e\phi.\mathbf{smile}(\mathbf{x}e)(\mathbf{y}e) \wedge \phi(\mathsf{upd}(\mathbf{smile}(\mathbf{x}e)(\mathbf{y}e), e)))\\
&(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x))\\
\rightarrow^*_\beta\ &\lambda e\phi.\mathbf{smile}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)e) \wedge\\
&\phi(\mathsf{upd}(\mathbf{smile}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)(\mathsf{sel}(\lambda x.\mathbf{female}x \wedge \mathbf{human}x)e), e)) && (5.41)
\end{aligned}
$$

Compare interpretation (5.41) with de Groote's [2006] interpretation (4.11), repeated in (5.42):

$$\lambda e\phi.\mathbf{smile}(\mathsf{sel}_{he}e)(\mathsf{sel}_{her}e) \wedge \phi e \qquad (5.42)$$

The terms are structurally similar. Interpretation (5.41) is, however, more realistic, because it provides the descriptive contents of pronouns as one of the arguments of function sel (e.g. (sel ($\lambda x.\mathbf{male}x \wedge \mathbf{human}x$) e)). Therefore, these descriptive contents can be used to retrieve the right individuals from the context. In (5.42), however, functions sel is simply marked (as a subscript) with the descriptive contents of pronouns and the descriptive contents are not provided to functions sel as arguments. Additionally, interpretation (5.41) updates the context (which is an argument of the term) with the proposition that he smiles at her. □

5.5 Donkey Sentences

Dynamic semantics examined in the previous sections successfully computes the meanings of donkey sentences compositionally. Consider, for example, Sentence (49):

(49) *Every farmer who owns a donkey beats it.*

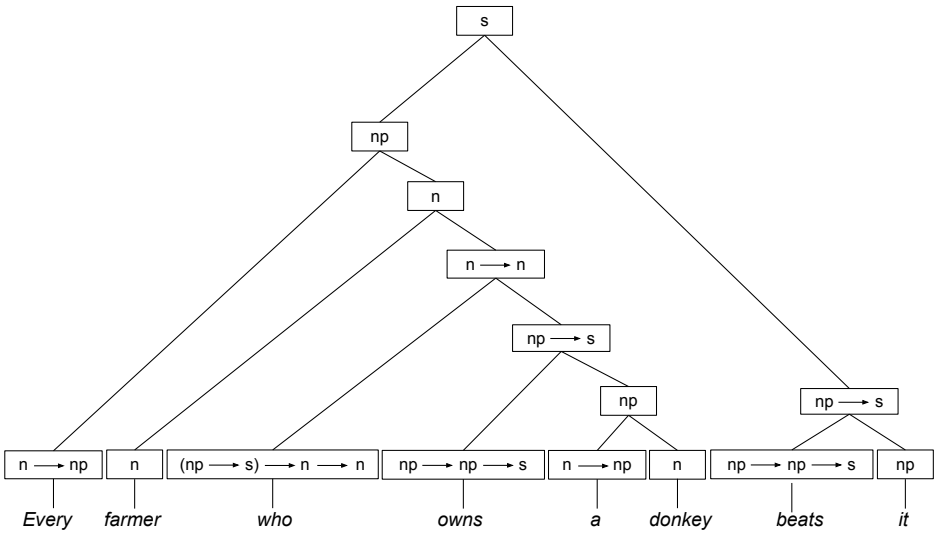


Figure 5.4: Syntactic parse tree of sentence *Every farmer who owns a donkey beats it.*

According to the parse tree, shown in Figure 2.1 and for convenience repeated in Figure 5.4, the dynamic meaning can be computed by β-reducing the term (5.43):

$$\overline{[\![beats]\!]}\ \widetilde{[\![it]\!]}(\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]})) \tag{5.43}$$

The dynamic lexical interpretations (in compact form) for words in Sentence (49) are shown in Table 5.4. These interpretations comply with dynamization principles discussed in Sections 5.1 and 5.2. Since the meanings of the lexical items are taken in compact form, the computation of the meaning of the sentence is free of "bureaucracy" and more intuitive compared to the computation using the normalized lexical interpretations.

The meaning of the noun phrase *a donkey* is computed by reducing the term $\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}$:

$$\begin{aligned}\overline{[\![a]\!]}\ \overline{[\![donkey]\!]} &= (\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px}\overline{\wedge}\mathbf{Qx}))\overline{[\![donkey]\!]}\\ &\rightarrow_\beta \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.\overline{[\![donkey]\!]}\mathbf{x}\ \overline{\wedge}\ \mathbf{Qx})\\ &= \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.\overline{\mathbf{dx}}\ \overline{\wedge}\ \mathbf{Qx})\end{aligned}$$

Lexical item	Syntactic category	Dynamic interpretation in GL
farmer	n	$\overline{\mathbf{f}}$
donkey	n	$\overline{\mathbf{d}}$
owns	$\mathrm{np} \to \mathrm{np} \to \mathrm{s}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{o}}\mathbf{xy}))$
beats	$\mathrm{np} \to \mathrm{np} \to \mathrm{s}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{b}}\mathbf{xy}))$
a	$\mathrm{n} \to \mathrm{np}$	$\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx})$
every	$\mathrm{n} \to \mathrm{np}$	$\lambda\mathbf{PQ}.\overline{\forall}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\Rightarrow}} \mathbf{Qx})$
who	$(\mathrm{np} \to \mathrm{s}) \to \mathrm{n} \to \mathrm{n}$	$\lambda\mathbf{RQx}.\mathbf{Qx} \mathbin{\overline{\wedge}} \mathbf{R}(\lambda\mathbf{P}.\mathbf{Px})$
it	np	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{non_human}x))$

Table 5.4: Dynamic interpretations of lexical items of the sentence *Every farmer who owns a donkey beats it* in framework GL.

The term $\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})$ is β-reduced as follows:

$$\begin{aligned}
\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}) &= (\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{o}}\mathbf{xy})))(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}) \\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}(\lambda\mathbf{y}.\overline{\mathbf{o}}\mathbf{xy})) \\
&= \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.(\lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \mathbf{Qz}))(\lambda\mathbf{y}.\overline{\mathbf{o}}\mathbf{xy})) \\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} (\lambda\mathbf{y}.\overline{\mathbf{o}}\mathbf{xy})\mathbf{z})) \\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{xz}))
\end{aligned}$$

The meaning of the relative clause *who owns a donkey* is computed by β-reducing the term $\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))$:

$$\begin{aligned}
\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})) &= (\lambda\mathbf{RQy}.\mathbf{Qy} \mathbin{\overline{\wedge}} \mathbf{R}(\lambda\mathbf{P}.\mathbf{Py}))(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})) \\
&\to_\beta \lambda\mathbf{Qy}.\mathbf{Qy} \mathbin{\overline{\wedge}} \overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})(\lambda\mathbf{P}.\mathbf{Py}) \\
&= \lambda\mathbf{Qy}.\mathbf{Qy} \mathbin{\overline{\wedge}} (\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{xz})))(\lambda\mathbf{P}.\mathbf{Py}) \\
&\to_\beta \lambda\mathbf{Qy}.\mathbf{Qy} \mathbin{\overline{\wedge}} (\lambda\mathbf{P}.\mathbf{Py})(\lambda\mathbf{x}.\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{xz})) \\
&\to_\beta \lambda\mathbf{Qy}.\mathbf{Qy} \mathbin{\overline{\wedge}} (\lambda\mathbf{x}.\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{xz})\mathbf{y}) \\
&\to_\beta \lambda\mathbf{Qy}.\mathbf{Qy} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{yz})
\end{aligned}$$

The meaning of *farmer who owns a donkey* is computed by applying the resulting λ-term in the computation above to the interpretation of *farmer*:

$$\begin{aligned}
(\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]} &= (\lambda\mathbf{Qy}.\mathbf{Qy} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{yz}))\overline{[\![farmer]\!]} \\
&\to_\beta \lambda\mathbf{y}.\overline{[\![farmer]\!]}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{yz}) \\
&= \lambda\mathbf{y}.\overline{\mathbf{f}}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{yz})
\end{aligned}$$

The meaning of the noun phrase *every farmer who owns a donkey* is computed by applying the interpretation of *every* to the interpretation of *farmer who owns*

a donkey:

$$\begin{aligned}
&\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]})\\
&= (\lambda\mathbf{PQ}.\overline{\forall}(\lambda\mathbf{x}.\mathbf{Px} \Rightarrow \mathbf{Qx}))((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]})\\
&\to_\beta \lambda\mathbf{Q}.\overline{\forall}(\lambda\mathbf{x}.((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]})\mathbf{x} \Rightarrow \mathbf{Qx})\\
&= \lambda\mathbf{Q}.\overline{\forall}(\lambda\mathbf{x}.(\lambda\mathbf{y}.\bar{\mathbf{f}}\mathbf{y} \barwedge \overline{\exists}(\lambda\mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \barwedge \bar{\mathbf{o}}\mathbf{yz}))\mathbf{x} \Rightarrow \mathbf{Qx})\\
&\to_\beta \lambda\mathbf{Q}.\overline{\forall}(\lambda\mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \barwedge \overline{\exists}(\lambda\mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \barwedge \bar{\mathbf{o}}\mathbf{xz})) \Rightarrow \mathbf{Qx}) \qquad (5.44)
\end{aligned}$$

The meaning of the verb phrase *beats it* is computed as follows:

$$\begin{aligned}
\overline{[\![beats]\!]}\ \widetilde{[\![it]\!]} &= (\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{xy})))\widetilde{[\![it]\!]}\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\widetilde{[\![it]\!]}(\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{xy}))\\
&= \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.(\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))(\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{xy}))\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.(\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{xy})(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\\
&\to_\beta \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \qquad (5.45)
\end{aligned}$$

Finally, the meaning of the sentence is computed by applying the interpretation (5.45) to the interpretation (5.44):

$$\begin{aligned}
&\overline{[\![beats]\!]}\ \widetilde{[\![it]\!]}(\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]}))\\
&= (\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))))\\
&\quad(\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]}))\\
&\to_\beta (\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]}))\\
&\quad(\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{y}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\\
&= (\lambda\mathbf{Q}.\overline{\forall}(\lambda\mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \barwedge \overline{\exists}(\lambda\mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \barwedge \bar{\mathbf{o}}\mathbf{xz})) \Rightarrow \mathbf{Qx}))(\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{y}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\\
&\to_\beta \overline{\forall}(\lambda\mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \barwedge \overline{\exists}(\lambda\mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \barwedge \bar{\mathbf{o}}\mathbf{xz})) \Rightarrow (\lambda\mathbf{y}.\bar{\mathbf{b}}\mathbf{y}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\mathbf{x})\\
&\to_\beta \overline{\forall}(\lambda\mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \barwedge \overline{\exists}(\lambda\mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \barwedge \bar{\mathbf{o}}\mathbf{xz})) \Rightarrow \bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \qquad (5.46)
\end{aligned}$$

Formula (5.46) is the resulting dynamic meaning of Sentence (49) in the compact form. Its computation is analogous to the computation of the static formula (2.15), and therefore more intuitive than if normalized (not compact, unfolded) interpretations of lexical items were used. Due to the similarity with the "static" computation, it may seem, at the first glance, that the formula (5.46) has the same problems as the static formula (2.15). However, this is not the case!

The "secret" of having the computation similar to and as intuitive as the static one but without the "static problems" lies in the fact that all the subterms used, particularly those standing for the logical connectives, are dynamic. Term (5.46) is analysed below in three ways. Firstly, simply by applying Proposition 4.35 to it; secondly, by examining its structure and its dynamic subterms; and finally, by normalizing it by performing step-by-step β-reductions.

Remember that by Remark 4.29 the following equations hold:

$$\overline{\forall} \doteq \lambda \mathbf{P}.\overline{\neg}\overline{\exists}(\overline{\neg}\mathbf{P}) \tag{5.47a}$$
$$\overline{\Rightarrow} \doteq \lambda \mathbf{A}\mathbf{B}.\overline{\neg}(\mathbf{A}\overline{\wedge}\overline{\neg}\mathbf{B}) \tag{5.47b}$$

Therefore, term (5.46) is equivalent to (5.48):

$$\overline{\neg}\overline{\exists}(\lambda \mathbf{x}.\bar{\mathbf{f}}\mathbf{x}\ \overline{\wedge}\ \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z}\ \overline{\wedge}\ \bar{\mathbf{o}}\mathbf{x}\mathbf{z})\ \overline{\wedge}\ \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \tag{5.48}$$

Analysis 1

Recall Proposition 4.35, stating that the following equation holds, provided $\mathbf{x} \notin FV(\overline{\mathrm{B}})$:

$$\overline{\exists}(\lambda \mathbf{x}.\overline{\mathrm{P}}\mathbf{x})\ \overline{\wedge}\ \overline{\mathrm{B}} =_\beta \overline{\exists}(\lambda \mathbf{x}.\overline{\mathrm{P}}\mathbf{x}\ \overline{\wedge}\ \overline{\mathrm{B}})$$

According to this proposition, since $\mathbf{z} \notin FV(\overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))$, formula (5.48) is β-equivalent to the following formula:

$$\overline{\neg}\overline{\exists}(\lambda \mathbf{x}.\bar{\mathbf{f}}\mathbf{x}\ \overline{\wedge}\ \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z}\ \overline{\wedge}\ \bar{\mathbf{o}}\mathbf{x}\mathbf{z}\ \overline{\wedge}\ \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))) \tag{5.49}$$

Finally, by Equations (5.47), formula (5.49) can be represented in a more familiar way (5.50):

$$\overline{\forall}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \overline{\Rightarrow} \overline{\forall}(\lambda \mathbf{z}.(\bar{\mathbf{d}}\mathbf{z}\ \overline{\wedge}\ \bar{\mathbf{o}}\mathbf{x}\mathbf{z}) \overline{\Rightarrow} \bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))))) \tag{5.50}$$

Analysis 2

Although it is sufficient to apply Proposition 4.35 to see that (5.46) is the desirable interpretation of the donkey sentence, it is worth to inspect the term itself (or its equivalent (5.48)) for gaining more intuition why this is so. Recall Equations (4.38e)–(4.38g) defining dynamic conjunction, existential quantifier and negation, repeated in (5.51):

$$\overline{\wedge} \doteq \lambda \mathbf{A}\mathbf{B}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi) \tag{5.51a}$$
$$\overline{\exists} \doteq \lambda \mathbf{P}.\lambda e\phi.\exists(\lambda x.\mathbf{P}(\lambda e.x)e\phi) \tag{5.51b}$$
$$\overline{\neg} \doteq \lambda \mathbf{A}.\lambda e\phi.\neg(\mathbf{A}e(\lambda e.\top)) \wedge \phi e \tag{5.51c}$$

First, it is important to see that the continuation of the subterm $\bar{\mathbf{f}}\mathbf{x}\ \overline{\wedge}\ \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z}\ \overline{\wedge}\ \bar{\mathbf{o}}\mathbf{x}\mathbf{z})$ of (5.48) is within the scope of the existential quantifier. For that, consider Formula (5.51a). The dynamic conjunction is defined in a way that its second argument $\mathbf{B}$ is passed as a subterm of the continuation $(\lambda e.\mathbf{B}e\phi)$ of the first argument $\mathbf{A}$. Therefore, in the subterm $\bar{\mathbf{d}}\mathbf{z}\ \overline{\wedge}\ \bar{\mathbf{o}}\mathbf{x}\mathbf{z}$, the term $\bar{\mathbf{o}}\mathbf{x}\mathbf{z}$ is

passed as the subterm of the continuation of $\overline{\mathbf{d}}\mathbf{z}$. Hence, the continuation ϕ of the term $\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}$ is deep inside the formula, as shown in (5.52)[10]:

$$\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z} \to^*_\beta \lambda e\phi.\mathbf{d}\mathbf{z}_ \wedge (\mathbf{o}(\mathbf{x}_)(\mathbf{z}_) \wedge \phi_) \tag{5.52}$$

Now, consider Equation (5.51b). $\overline{\exists}$ applied to $\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}$ results in a formula that has the form shown in (5.53). Note that the continuation remains deep inside the formula and, what is the most important, it is within the scope of the existential quantifier:

$$\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}) \to^*_\beta \lambda e\phi.\exists(\lambda z.\mathbf{d}z \wedge (\mathbf{o}(\mathbf{x}_)z \wedge \phi_)) \tag{5.53}$$

Now the dynamic conjunction of $\bar{\mathbf{f}}\mathbf{x}$ and $\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z})$ can be analysed. By Definition (5.51a), the whole term $\overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z})$ goes as a subterm of the continuation of $\bar{\mathbf{f}}\mathbf{x}$, resulting in the term having the form shown in (5.54). Importantly, the continuation ϕ of the term is deep inside the formula and under the scope of the existential quantifier:

$$\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}) \to^*_\beta \lambda e\phi.\mathbf{f}(\mathbf{x}_) \wedge \exists(\lambda z.\mathbf{d}z \wedge (\mathbf{o}(\mathbf{x}_)z \wedge \phi_)) \tag{5.54}$$

Returning to term (5.48), the subterm $(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}))$ is dynamically conjoint with $\overline{\neg}\overline{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))$. Hence, by Definition (5.51a), the term $\overline{\neg}\overline{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))$ is passed as the subterm of the continuation of the term $(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}))$. As shown in (5.54), the continuation ϕ is under the scope of the existential quantifier, and, therefore, this is the very moment when the formula interpreting *beats it* goes under the scope of the existential quantifier that binds the variable interpreting *a donkey*, resulting in the term having the form (5.55):

$$\begin{aligned}&(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\overline{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))\\ \to^*_\beta\ &\lambda e\phi.\mathbf{f}(\mathbf{x}_) \wedge \exists(\lambda z.\mathbf{d}z \wedge (\mathbf{o}(\mathbf{x}_)z \wedge (\neg\mathbf{b}(\mathbf{x}_)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)_) \wedge \phi_)))\end{aligned} \tag{5.55}$$

The fact that, due to the definition of $\overline{\wedge}$, $\overline{\neg}\overline{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))$ entered deep inside the formula $(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda\mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}))$ is not only relevant for binding *it* with the quantifier of *a donkey*, but is also the reason for the existential quantifier to change to an essentially universal quantifier, since it is moved to a position with negative polarity. In other words, abbreviating $\mathbf{f}(\mathbf{x}_)$ by $A(x)$ and $\exists(\lambda z.\mathbf{d}z \wedge (\mathbf{o}(\mathbf{x}_)z \wedge (\neg\mathbf{b}(\mathbf{x}_)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)_) \wedge \phi_)))$ by $\exists z.G(z)$, formula (5.49) is of the form (5.56), which is logically equivalent to (5.57):

$$\neg\exists x.(A(x) \wedge \exists z.G(z)) \tag{5.56}$$

$$\equiv \forall x.(A(x) \to \forall z.\neg G(z)) \tag{5.57}$$

Therefore, during the normalization of the term (5.46), its dynamic subterms do the right job, and the formula interpreting *beats it* appears under the scope of the universal quantifier binding the variable interpreting *a donkey*.

[10] Here and below the underline _ indicates some omitted terms.

Analysis 3

Below the normalization of the formula (5.48) is conducted in detail. In order to reduce the size of the λ-terms in the computations below, the following notation is used:

NOTATION 5.11. An environment e updated with a term F of type o, i.e. $\mathsf{upd}(F, e)$, is denoted as $(F \bullet e)$. Operation $\bullet$ is right associative. For example, $(F_1 \bullet F_2 \bullet e)$ is equivalent to $(F_1 \bullet (F_2 \bullet e))$.

As in previous examples upd (or $\bullet$) can be seen as a conjunction of its arguments. However, the symbol $\bullet$ is used for a better visualisation of terms by avoiding the abundance of symbol $\wedge$ (note that $\bullet$ appears in environment only).
The dynamic interpretations of constants are as follows:

$$\bar{\mathbf{f}} = \lambda \mathbf{a} e \phi.\mathbf{f}(\mathbf{a}e) \wedge \phi(\mathbf{f}(\mathbf{a}e) \bullet e) \tag{5.58a}$$

$$\overline{\mathbf{d}} = \lambda \mathbf{a} e \phi.\mathbf{d}(\mathbf{a}e) \wedge \phi(\mathbf{d}(\mathbf{a}e) \bullet e) \tag{5.58b}$$

$$\overline{\mathbf{o}} = \lambda \mathbf{a}_1 \mathbf{a}_2 . \lambda e \phi.\mathbf{o}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \wedge \phi(\mathbf{o}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \bullet e) \tag{5.58c}$$

$$\overline{\mathbf{b}} = \lambda \mathbf{a}_1 \mathbf{a}_2 . \lambda e \phi.\mathbf{b}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \wedge \phi(\mathbf{b}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \bullet e) \tag{5.58d}$$

In what follows the normalization of (5.48) is performed by consecutive applications of its respective subterms:

$$\begin{aligned}
&\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z} \\
&= (\lambda \mathbf{a} e \phi.\mathbf{d}(\mathbf{a}e) \wedge \phi(\mathbf{d}(\mathbf{a}e) \bullet e))\mathbf{z} \mathbin{\overline{\wedge}} (\lambda \mathbf{a}_1 \mathbf{a}_2 . \lambda e \phi.\mathbf{o}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \ \wedge \ \phi(\mathbf{o}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \bullet e))\mathbf{x}\mathbf{z} \\
&\to_\beta^* (\lambda e \phi.\mathbf{d}(\mathbf{z}e) \wedge \phi(\mathbf{d}(\mathbf{z}e) \bullet e)) \mathbin{\overline{\wedge}} (\lambda e \phi.\mathbf{o}(\mathbf{x}e)(\mathbf{z}e) \wedge \phi(\mathbf{o}(\mathbf{x}e)(\mathbf{z}e) \bullet e)) \\
&\to_\beta^* \lambda e' \phi'.(\lambda e \phi.\mathbf{d}(\mathbf{z}e) \wedge \phi(\mathbf{d}(\mathbf{z}e) \bullet e))e'(\lambda e''.(\lambda e \phi.\mathbf{o}(\mathbf{x}e)(\mathbf{z}e) \wedge \phi(\mathbf{o}(\mathbf{x}e)(\mathbf{z}e) \bullet e))e'' \phi') \\
&\to_\beta^* \lambda e' \phi'.(\lambda e \phi.\mathbf{d}(\mathbf{z}e) \wedge \phi(\mathbf{d}(\mathbf{z}e) \bullet e))e'(\lambda e''.\mathbf{o}(\mathbf{x}e'')(\mathbf{z}e'') \wedge \phi'(\mathbf{o}(\mathbf{x}e'')(\mathbf{z}e'') \bullet e'')) \\
&\to_\beta \lambda e' \phi'.(\lambda \phi.\mathbf{d}(\mathbf{z}e') \wedge \phi(\mathbf{d}(\mathbf{z}e') \bullet e'))(\lambda e''.\mathbf{o}(\mathbf{x}e'')(\mathbf{z}e'') \wedge \phi'(\mathbf{o}(\mathbf{x}e'')(\mathbf{z}e'') \bullet e'')) \\
&\to_\beta \lambda e' \phi'.\mathbf{d}(\mathbf{z}e') \wedge (\lambda e''.\mathbf{o}(\mathbf{x}e'')(\mathbf{z}e'') \wedge \phi'(\mathbf{o}(\mathbf{x}e'')(\mathbf{z}e'') \bullet e''))(\mathbf{d}(\mathbf{z}e') \bullet e') \\
&\to_\beta \lambda e' \phi'.\mathbf{d}(\mathbf{z}e') \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}(\mathbf{z}e') \bullet e'))(\mathbf{z}(\mathbf{d}(\mathbf{z}e') \bullet e')) \wedge \\
&\qquad \phi'(\mathbf{o}(\mathbf{x}(\mathbf{d}(\mathbf{z}e') \bullet e'))(\mathbf{z}(\mathbf{d}(\mathbf{z}e') \bullet e')) \bullet \mathbf{d}(\mathbf{z}e') \bullet e'))
\end{aligned} \tag{5.59}$$

Compare (5.59) to (5.52).

$$\begin{aligned}
&\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}) \\
&= (\lambda \mathbf{P} e \phi.\exists(\lambda y.\mathbf{P}(\lambda e.y)(e)\phi))(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}) \\
&\to_\beta \lambda e \phi.\exists(\lambda y.(\lambda \mathbf{z}.(\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \overline{\mathbf{o}}\mathbf{x}\mathbf{z}))(\lambda e.y)(e)\phi) \\
&= \lambda e \phi.\exists(\lambda y.(\lambda \mathbf{z}.(\lambda e' \phi'.\mathbf{d}(\mathbf{z}e') \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}(\mathbf{z}e') \bullet e'))(\mathbf{z}(\mathbf{d}(\mathbf{z}e') \bullet e')) \wedge \\
&\qquad \phi'(\mathbf{o}(\mathbf{x}(\mathbf{d}(\mathbf{z}e') \bullet e'))(\mathbf{z}(\mathbf{d}(\mathbf{z}e') \bullet e')) \bullet \mathbf{d}(\mathbf{z}e') \bullet e'))))(\lambda e.y)(e)\phi) \\
&\to_\beta^* \lambda e \phi.\exists(\lambda y.(\lambda e' \phi'.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e'))y \wedge \phi'(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e'))y \bullet \mathbf{d}y \bullet e')))(e)\phi) \\
&\to_\beta^* \lambda e \phi.\exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e))y \wedge \phi(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e))y \bullet \mathbf{d}y \bullet e)))
\end{aligned} \tag{5.60}$$

Compare (5.60) to (5.53).

$$
\begin{aligned}
&\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}) \\
&= (\lambda \mathbf{a} e \phi.\mathbf{f}(\mathbf{a}e) \wedge \phi(\mathbf{f}(\mathbf{a}e) \bullet e))\mathbf{x} \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}) \\
&\to_\beta (\lambda e \phi.\mathbf{f}(\mathbf{x}e) \wedge \phi(\mathbf{f}(\mathbf{x}e) \bullet e)) \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}) \\
&= (\lambda e \phi.\mathbf{f}(\mathbf{x}e) \wedge \phi(\mathbf{f}(\mathbf{x}e) \bullet e)) \mathbin{\overline{\wedge}} (\lambda e \phi.\exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e))y \wedge \\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e))y \bullet \mathbf{d}y \bullet e)))) \\
&= \lambda e \phi.(\lambda e \phi.\mathbf{f}(\mathbf{x}e) \wedge \phi(\mathbf{f}(\mathbf{x}e) \bullet e))e(\lambda e'.(\lambda e \phi.\exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e))y \wedge \\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e))y \bullet \mathbf{d}y \bullet e)))e'\phi) \\
&\to_\beta^* \lambda e \phi.(\lambda e \phi.\mathbf{f}(\mathbf{x}e) \wedge \phi(\mathbf{f}(\mathbf{x}e) \bullet e))e(\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e'))y \wedge \\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e'))y \bullet \mathbf{d}y \bullet e')))) \\
&\to_\beta^* \lambda e \phi.\mathbf{f}(\mathbf{x}e) \wedge (\lambda e'.\exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e'))y \wedge \\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet e'))y \bullet \mathbf{d}y \bullet e'))))(\mathbf{f}(\mathbf{x}e) \bullet e) \\
&\to_\beta \lambda e \phi.\mathbf{f}(\mathbf{x}e) \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet \mathbf{f}(\mathbf{x}e) \bullet e))y \wedge \\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}(\mathbf{d}y \bullet \mathbf{f}(\mathbf{x}e) \bullet e))y \bullet \mathbf{d}y \bullet \mathbf{f}(\mathbf{x}e) \bullet e)))
\end{aligned} \tag{5.61}
$$

Compare (5.61) to (5.54).

$$
\begin{aligned}
&\bar{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)) \\
&= \bar{\neg}(\lambda \mathbf{a}_1 \mathbf{a}_2.\lambda e \phi.\mathbf{b}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \wedge \phi(\mathbf{b}(\mathbf{a}_1 e)(\mathbf{a}_2 e) \bullet e))\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)) \\
&\to_\beta \bar{\neg}(\lambda e \phi.\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \phi(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \bullet e)) \\
&= (\lambda \mathbf{A}.\lambda e \phi.\neg(\mathbf{A}e(\lambda e.\top)) \wedge \phi e)(\lambda e \phi.\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \\
&\qquad\qquad \phi(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \bullet e)) \\
&\to_\beta \lambda e \phi.\neg((\lambda e \phi.(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \\
&\qquad\qquad \phi(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \bullet e))e(\lambda e.\top)) \wedge \phi e \\
&\to_\beta^* \lambda e \phi.\neg(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \\
&\qquad\qquad (\lambda e.\top)(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \bullet e)) \wedge \phi e \\
&\to_\beta \lambda e \phi.\neg(\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \top) \wedge \phi e \\
&\equiv \lambda e \phi.\neg\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \phi e
\end{aligned} \tag{5.62}
$$

In the computations below e_1 is used as an abbreviation for $(\mathbf{d}y \bullet \mathbf{f}(\mathbf{x}e) \bullet e)$:

$$
\begin{aligned}
&(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \bar{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)) \\
&= (\lambda \mathbf{A}\mathbf{B}.\lambda e \phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi))(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}))(\bar{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \\
&\to_\beta^* \lambda e \phi.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}))e(\lambda e.(\bar{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))e\phi) \\
&= \lambda e \phi.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \bar{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}))e(\lambda e.(\lambda e \phi.\neg\mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \phi e)e\phi)
\end{aligned}
$$

$$
\begin{aligned}
&\to^*_\beta \lambda e\phi.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}))e(\lambda e.\neg \mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \phi e)\\
&\quad = \lambda e\phi.(\lambda e\phi.\mathbf{f}(\mathbf{x}e) \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}e_1)y \wedge \phi(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1))))\\
&\qquad\quad e(\lambda e.\neg \mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \phi e)\\
&\to^*_\beta \lambda e\phi.\mathbf{f}(\mathbf{x}e) \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}e_1)y \wedge (\lambda e.\neg \mathbf{b}(\mathbf{x}e)(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e) \wedge \phi e)\\
&\qquad\qquad\qquad\qquad\qquad\qquad (\mathbf{o}(\mathbf{x}e_1)y \bullet e_1)))\\
&\to_\beta \lambda e\phi.\mathbf{f}(\mathbf{x}e) \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}e_1)y \wedge\\
&\qquad\quad (\neg \mathbf{b}(\mathbf{x}(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1))(\mathsf{sel}(\lambda z.\mathbf{non_human}z)(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1)) \wedge\\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1))))
\end{aligned}
\tag{5.63}
$$

Compare (5.63) to (5.55). Below e_2 abbreviates $(\mathbf{o}xy \bullet \mathbf{d}y \bullet \mathbf{f}x \bullet e)$:

$$
\begin{aligned}
&\overline{\exists}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\\
&= (\lambda \mathbf{P}.\lambda e\phi.\exists(\lambda x.\mathbf{P}(\lambda e.x)(e)\phi))\\
&\qquad (\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\\
&\to_\beta \lambda e\phi.\exists(\lambda x.(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))(\lambda e.x)(e)\phi)\\
&= \lambda e\phi.\exists(\lambda x.(\lambda \mathbf{x}.(\lambda e\phi.\mathbf{f}(\mathbf{x}e) \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}(\mathbf{x}e_1)y \wedge\\
&\qquad\quad (\neg \mathbf{b}(\mathbf{x}(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1))(\mathsf{sel}(\lambda z.\mathbf{non_human}z)(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1)) \wedge\\
&\qquad\qquad \phi(\mathbf{o}(\mathbf{x}e_1)y \bullet e_1))))))(\lambda e.x)(e)\phi)\\
&\to^*_\beta \lambda e\phi.\exists(\lambda x.(\lambda e\phi.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)(\mathbf{o}xy \bullet e_1)) \wedge\\
&\qquad\quad \phi(\mathbf{o}xy \bullet e_1))))))(e)\phi)\\
&\to_\beta \lambda e\phi.\exists(\lambda x.(\lambda e\phi.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge\\
&\qquad\quad (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_2) \wedge \phi e_2))))(e)\phi)\\
&\to^*_\beta \lambda e\phi.\exists(\lambda x.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_2) \wedge \phi e_2))))
\end{aligned}
\tag{5.64}
$$

$$
\begin{aligned}
&\overline{\neg}\overline{\exists}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\\
&= (\lambda \mathbf{A}.\lambda e\phi.\neg(\mathbf{A}e(\lambda e.\top)) \wedge \phi e)\\
&\qquad (\overline{\exists}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))))\\
&\to_\beta \lambda e\phi.\neg((\overline{\exists}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \mathbin{\overline{\wedge}} \overline{\neg}\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))))e(\lambda e.\top)) \wedge \phi e\\
&= \lambda e\phi.\neg((\lambda e\phi.\exists(\lambda x.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge\\
&\qquad\quad (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_2) \wedge \phi e_2)))))e(\lambda e.\top)) \wedge \phi e\\
&\to^*_\beta \lambda e\phi.\neg\exists(\lambda x.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_2) \wedge (\lambda e.\top)e_2)))) \wedge \phi e\\
&\to_\beta \lambda e\phi.\neg\exists(\lambda x.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_2) \wedge \top)))) \wedge \phi e\\
&\equiv \lambda e\phi.\neg\exists(\lambda x.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}xy \wedge (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_2))))) \wedge \phi e\\
&= \lambda e\phi.\neg\exists(\lambda x.\mathbf{f}x \wedge \exists(\lambda y.\mathbf{d}y \wedge (\mathbf{o}xy \wedge\\
&\qquad\quad (\neg \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)(\mathbf{o}xy \bullet \mathbf{d}y \bullet \mathbf{f}x \bullet e)))))) \wedge \phi e
\end{aligned}
\tag{5.65}
$$

Equation (5.65) is the normal form of the dynamic interpretation of the sentence. It is logically equivalent to (5.66):

$$\lambda e\phi.\forall(\lambda x.\mathbf{f}x \rightarrow \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \rightarrow \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)(\mathbf{o}xy \bullet \mathbf{d}y \bullet \mathbf{f}x \bullet e)))) \wedge \phi e \quad (5.66)$$

Comparison of the Static and the Compact Dynamic Computations of Meaning

The last two pages in this chapter show a side-by-side comparison of the compositional static computation of the meaning of the donkey sentence, discussed in Subsection 2.1.3, and the compact dynamic computation in framework GL discussed above. The comparison clearly shows that the dynamic computation is analogous to and as intuitive as the static one. However, as has been shown above, the dynamic computation overcomes the challenges faced by the static computation.

Static interpretation

$$\begin{aligned}
[\![a]\!][\![donkey]\!] &= (\lambda \mathrm{PQ}.\exists(\lambda x.\mathrm{P}x \wedge \mathrm{Q}x))[\![donkey]\!] \\
&\rightarrow_\beta \lambda \mathrm{Q}.\exists(\lambda x.[\![donkey]\!]x \wedge \mathrm{Q}x) \\
&= \lambda \mathrm{Q}.\exists(\lambda x.\mathbf{d}x \wedge \mathrm{Q}x)
\end{aligned}$$

$$\begin{aligned}
[\![owns]\!]([\![a]\!][\![donkey]\!]) &= (\lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{o}xy)))([\![a]\!][\![donkey]\!]) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.[\![a]\!][\![donkey]\!](\lambda y.\mathbf{o}xy)) \\
&= \lambda \mathrm{X}.\mathrm{X}(\lambda x.(\lambda \mathrm{Q}.\exists(\lambda z.\mathbf{d}z \wedge \mathrm{Q}z))(\lambda y.\mathbf{o}xy)) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge (\lambda y.\mathbf{o}xy)z)) \\
&\rightarrow_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz))
\end{aligned}$$

$$\begin{aligned}
&[\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])) \\
&= (\lambda \mathrm{RQ}y.\mathrm{Q}y \wedge \mathrm{R}(\lambda \mathrm{P}.\mathrm{P}y))([\![owns]\!]([\![a]\!][\![donkey]\!])) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge [\![owns]\!]([\![a]\!][\![donkey]\!])(\lambda \mathrm{P}.\mathrm{P}y) \\
&= \lambda \mathrm{Q}y.\mathrm{Q}y \wedge (\lambda \mathrm{X}.\mathrm{X}(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)))(\lambda \mathrm{P}.\mathrm{P}y) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge (\lambda \mathrm{P}.\mathrm{P}y)(\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge (\lambda x.\exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)y) \\
&\rightarrow_\beta \lambda \mathrm{Q}y.\mathrm{Q}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz)
\end{aligned}$$

$$\begin{aligned}
&([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!])))[\![farmer]\!] \\
&= (\lambda \mathrm{Q}y.\mathrm{Q}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz))[\![farmer]\!] \\
&\rightarrow_\beta \lambda y.[\![farmer]\!]y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz) \\
&= \lambda y.\mathbf{f}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz)
\end{aligned}$$

Dynamic interpretation

$$\begin{aligned}
\overline{[\![a]\!]}\ \overline{[\![donkey]\!]} &= (\lambda \mathbf{PQ}.\overline{\exists}(\lambda \mathbf{x}.\mathbf{P}\mathbf{x}\overline{\wedge}\mathbf{Q}\mathbf{x}))\overline{[\![donkey]\!]} \\
&\rightarrow_\beta \lambda \mathbf{Q}.\overline{\exists}(\lambda \mathbf{x}.\overline{[\![donkey]\!]}\mathbf{x} \mathbin{\overline{\wedge}} \mathbf{Q}\mathbf{x}) \\
&= \lambda \mathbf{Q}.\overline{\exists}(\lambda \mathbf{x}.\overline{\mathbf{d}}\mathbf{x} \mathbin{\overline{\wedge}} \mathbf{Q}\mathbf{x})
\end{aligned}$$

$$\begin{aligned}
\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}) &= (\lambda \mathbf{YX}.\mathbf{X}(\lambda \mathbf{x}.\mathbf{Y}(\lambda \mathbf{y}.\bar{\mathbf{o}}\mathbf{x}\mathbf{y})))(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}) \\
&\rightarrow_\beta \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}(\lambda \mathbf{y}.\bar{\mathbf{o}}\mathbf{x}\mathbf{y})) \\
&= \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.(\lambda \mathbf{Q}.\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \mathbf{Q}\mathbf{z}))(\lambda \mathbf{y}.\bar{\mathbf{o}}\mathbf{x}\mathbf{y})) \\
&\rightarrow_\beta \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} (\lambda \mathbf{y}.\bar{\mathbf{o}}\mathbf{x}\mathbf{y})\mathbf{z})) \\
&\rightarrow_\beta \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z}))
\end{aligned}$$

$$\begin{aligned}
&\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})) \\
&= (\lambda \mathbf{RQy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} \mathbf{R}(\lambda \mathbf{P}.\mathbf{P}\mathbf{y}))(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})) \\
&\rightarrow_\beta \lambda \mathbf{Qy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} \overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})(\lambda \mathbf{P}.\mathbf{P}\mathbf{y}) \\
&= \lambda \mathbf{Qy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} (\lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})))(\lambda \mathbf{P}.\mathbf{P}\mathbf{y}) \\
&\rightarrow_\beta \lambda \mathbf{Qy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} (\lambda \mathbf{P}.\mathbf{P}\mathbf{y})(\lambda \mathbf{x}.\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \\
&\rightarrow_\beta \lambda \mathbf{Qy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} (\lambda \mathbf{x}.\overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})\mathbf{y}) \\
&\rightarrow_\beta \lambda \mathbf{Qy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{y}\mathbf{z})
\end{aligned}$$

$$\begin{aligned}
&(\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]})))\overline{[\![farmer]\!]} \\
&= (\lambda \mathbf{Qy}.\mathbf{Q}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{y}\mathbf{z}))\overline{[\![farmer]\!]} \\
&\rightarrow_\beta \lambda \mathbf{y}.\overline{[\![farmer]\!]}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{y}\mathbf{z}) \\
&= \lambda \mathbf{y}.\bar{\mathbf{f}}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\overline{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{y}\mathbf{z})
\end{aligned}$$

$$
\begin{aligned}
&[\![every]\!]([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!]) \\
&= (\lambda \mathrm{PQ}.\forall(\lambda x.\mathrm{P}x \to \mathrm{Q}x))(([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!]) \\
&\to_\beta \lambda \mathrm{Q}.\forall(\lambda x.([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!]x \to \mathrm{Q}x) \\
&= \lambda \mathrm{Q}.\forall(\lambda x.(\lambda y.\mathbf{f}y \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}yz))x \to \mathrm{Q}x) \\
&\to_\beta \lambda \mathrm{Q}.\forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \to \mathrm{Q}x)
\end{aligned}
$$

$$
\begin{aligned}
&\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))\overline{[\![farmer]\!]}) \\
&= (\lambda \mathbf{PQ}.\overline{\forall}(\lambda \mathbf{x}.\mathbf{P}\mathbf{x} \Rightarrow \mathbf{Q}\mathbf{x}))((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))\overline{[\![farmer]\!]}) \\
&\to_\beta \lambda \mathbf{Q}.\overline{\forall}(\lambda \mathbf{x}.((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))\overline{[\![farmer]\!]})\mathbf{x} \Rightarrow \mathbf{Q}\mathbf{x}) \\
&= \lambda \mathbf{Q}.\overline{\forall}(\lambda \mathbf{x}.(\lambda \mathbf{y}.\bar{\mathbf{f}}\mathbf{y} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{y}\mathbf{z}))\mathbf{x} \Rightarrow \mathbf{Q}\mathbf{x}) \\
&\to_\beta \lambda \mathbf{Q}.\overline{\forall}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \Rightarrow \mathbf{Q}\mathbf{x})
\end{aligned}
$$

$$
\begin{aligned}
[\![beats]\!][\![it]\!] &= (\lambda \mathrm{YX}.\mathrm{X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{b}xy)))[\![it]\!] \\
&\to_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.[\![it]\!](\lambda y.\mathbf{b}xy)) \\
&= \lambda \mathrm{X}.\mathrm{X}(\lambda x.(\lambda \mathrm{P}.\mathrm{P}\wp^?)(\lambda y.\mathbf{b}xy)) \\
&\to_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.(\lambda y.\mathbf{b}xy)\wp^?) \\
&\to_\beta \lambda \mathrm{X}.\mathrm{X}(\lambda x.\mathbf{b}x\wp^?)
\end{aligned}
$$

$$
\begin{aligned}
\overline{[\![beats]\!]}\ \widetilde{[\![it]\!]} &= (\lambda \mathbf{YX}.\mathbf{X}(\lambda \mathbf{x}.\mathbf{Y}(\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{x}\mathbf{y})))\widetilde{[\![it]\!]} \\
&\to_\beta \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\widetilde{[\![it]\!]}(\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{x}\mathbf{y})) \\
&= \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.(\lambda \mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))(\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{x}\mathbf{y})) \\
&\to_\beta \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.(\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{x}\mathbf{y})(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \\
&\to_\beta \lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))
\end{aligned}
$$

$$
\begin{aligned}
&[\![beats]\!][\![it]\!]([\![every]\!]([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!])) \\
&= (\lambda \mathrm{X}.\mathrm{X}(\lambda x.\mathbf{b}x\wp^?))([\![every]\!]([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!])) \\
&\to_\beta ([\![every]\!]([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!]))(\lambda x.\mathbf{b}x\wp^?) \\
&= (\lambda \mathrm{Q}.\forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \to \mathrm{Q}x))(\lambda x.\mathbf{b}x\wp^?) \\
&\to_\beta \forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \to (\lambda x.\mathbf{b}x\wp^?)x) \\
&\to_\beta \forall(\lambda x.(\mathbf{f}x \wedge \exists(\lambda z.\mathbf{d}z \wedge \mathbf{o}xz)) \to \mathbf{b}x\wp^?)
\end{aligned}
$$

$$
\begin{aligned}
&\overline{[\![beats]\!]}\ \widetilde{[\![it]\!]}(\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))\overline{[\![farmer]\!]})) \\
&= (\lambda \mathbf{X}.\mathbf{X}(\lambda \mathbf{x}.\bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))))(\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))\overline{[\![farmer]\!]})) \\
&\to_\beta (\overline{[\![every]\!]}((\overline{[\![who]\!]}(\overline{[\![owns]\!]}(\overline{[\![a]\!]}\ \overline{[\![donkey]\!]}))\overline{[\![farmer]\!]}))(\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{y}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \\
&= (\lambda \mathbf{Q}.\overline{\forall}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \Rightarrow \mathbf{Q}\mathbf{x}))(\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{y}(\mathsf{sel}(\lambda z.\mathbf{non_human}z))) \\
&\to_\beta \overline{\forall}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \Rightarrow (\lambda \mathbf{y}.\bar{\mathbf{b}}\mathbf{y}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))\mathbf{x}) \\
&\to_\beta \overline{\forall}(\lambda \mathbf{x}.(\bar{\mathbf{f}}\mathbf{x} \mathbin{\overline{\wedge}} \overline{\exists}(\lambda \mathbf{z}.\bar{\mathbf{d}}\mathbf{z} \mathbin{\overline{\wedge}} \bar{\mathbf{o}}\mathbf{x}\mathbf{z})) \Rightarrow \bar{\mathbf{b}}\mathbf{x}(\mathsf{sel}(\lambda z.\mathbf{non_human}z)))
\end{aligned}
$$

This chapter demonstrated how higher order continuation-based dynamic framework GL developed in Section 4.3.1 can be applied for giving interpretations to natural language lexical constituents and for compositionally computing meanings of sentences from the lexical meanings. To express unconventional context-dependent meaning a mechanism of exception raising is used. Thus, referring expressions are interpreted with terms that can raise exceptions when evaluated with respect to a context that does not contain a referent satisfying the descriptive content of the expressions (this corresponds to presuppositions triggering). Exceptions are raised and handled when a sentence updates some preceding discourse. The next chapter develops such a discourse update with exception handling mechanism.

Chapter 6

Continuation-based Dynamic Logic with Exceptions: Interpretation of Discourse

The principle of compositionality, discussed in Section 1.2, is a fundamental requirement necessary to finitely express an infinite number of possible meanings. Chapter 2 shows that it is not enough to compositionally compute the meanings of sentences: due to the possible correlation between sentences, the meaning of discourse, composed of those sentences, should be also computed compositionally. This chapter defines a function that takes an interpretation of a discourse and an interpretation of a sentence and returns an interpretation of an updated discourse. This function uses exception handling mechanism for accommodating presuppositions triggered by referring expressions when the context of the discourse does not contain appropriate referents. The use of exception raising and handling mechanism motivates the extension of the calculus of framework GL, resulting in framework GLχ.

6.1 Discourse Update: Accommodation as Exception Handling

A discourse is a sequence of sentences. Therefore, given separate interpretations of sentences, as, for example, interpretations (5.34), (5.37) and (5.41) for sentences (50a), (50b) and (50c) respectively, it is desirable to compositionally compute the interpretation of the discourse consisting of these sentences, as discourse (50d):

(50) a. *The story is about John.*

b. *John loves Mary.*

c. *He smiles at her.*

d. *The story is about John. John loves Mary. He smiles at her.*

Since the discourse is a sequence of sentences, the computation of the meaning of discourse has to be done incrementally, starting from the first sentence in the order as the sentences appear. Consider that the initial part of some discourse has already been interpreted (i.e. an interpretation $\mathbf{D}_i$ from the first sentence of the discourse up to the i'th sentence (including it) is already computed). For a new $(i+1)$'th sentence this interpretation should be updated with the interpretation $\mathbf{S}_{i+1}$ of this sentence, resulting in the interpretation $\mathbf{D}_{i+1}$ of a new discourse that contains the semantic contents of the discourse processed so far and of the $(i+1)$'th sentence (i.e. from the first to the $(i+1)$'th sentences) and, moreover, the context, in which the pragmatical issues caused by the $(i+1)$'th sentence are resolved.

Recall that interpretations of sentences are terms of type (6.1), the type of a dynamic proposition:

$$[\![s]\!] = \gamma \to (\gamma \to o) \to o \tag{6.1}$$

This means that the interpretation of a sentence takes a context (of type γ) as one of its arguments and a continuation (of type $(\gamma \to o)$) as another argument. Informally, one can think of the meaning of an individual sentence as something that has to be evaluated with respect to a context to be fully understood. A discourse, on the other hand, can be seen as something that already has a concrete context. Therefore, interpretations of discourses can be defined as the terms of type (6.2):

$$[\![d]\!] = (\gamma \to o) \to o \tag{6.2}$$

Thus, the current discourse is represented by a term that takes a continuation as its only argument, which is necessary for the computation of the meaning of the remainder of the discourse. Every new sentence updates the discourse and its context. Then a tentative discourse update function dupd, which takes the meaning of the current discourse and the meaning of the new sentence as arguments, can be defined as follows:

Definition 6.1. [Discourse update (preliminary)] Let $\mathbf{D}$ be a term of type $((\gamma \to o) \to o)$ and $\mathbf{S}$ be a term of type $(\gamma \to (\gamma \to o) \to o)$. Then the function dupd is defined by the following equation:

$$\mathsf{dupd}\ \mathbf{D}\ \mathbf{S} \doteq \lambda\phi.\mathbf{D}(\lambda e.\mathbf{S}e\phi) \tag{6.3}$$

The update of the interpretation $\mathbf{D}$ of the current discourse with the interpretation $\mathbf{S}$ of a new sentence results in the interpretation of the discourse containing

the new sentence (as its last sentence). Therefore, the resulting term has to be of type $((\gamma \to o) \to o)$. Hence, the term should have only one argument of type $(\gamma \to o)$. This explains the abstraction over the variable ϕ in (6.3). The body of the term must be a term of type o contributed by **D** and **S**. **D** should go first in the logical formula. It requires one argument of type $(\gamma \to o)$, which should be constructed with **S** of type $(\gamma \to (\gamma \to o) \to o)$. This explains the subterm $(\lambda e.\mathbf{S}e\phi)$. Importantly, $(\lambda e.\mathbf{S}e\phi)$ functions as a continuation of **D**. Therefore, the context of **D** is available for **S** (and **S** can update the context!). Indeed, the context of **D** is passed as the first argument of **S** due to the fact that the subterm $(\lambda e.\mathbf{S}e\phi)$ is an abstraction.

Definition 6.2. [Interpretation of initial discourse] Let c be a context. Then the initial (content-empty) discourse $\mathbf{D}_0$ containing context c is interpreted as the following term:

$$\mathbf{D}_0 \doteq \lambda\phi.\phi c \tag{6.4}$$

A realistic context c should contain common knowledge and information learned since an agent interpreting the discourse was put into functioning. For simplification, however, context c in the initial discourse can be considered either empty or containing only a necessary part of some knowledge. By letting context be a proposition (i.e. by letting γ be o), the empty context can be represented by $\top$.[1]

Example 6.3. Consider the following formulas representing a tiny part of common knowledge. Formula (6.5a) expresses the fact that whoever is named "John" or "Mary" is a human. Formula (6.5b) expresses the fact that whoever is named "John" is male. Formula (6.5c) expresses the fact that whoever is called "Mary" is female.

$$\forall x.(\text{named "John"}x) \vee (\text{named "Mary"}x) \to \mathbf{human}x \tag{6.5a}$$
$$\forall x.(\text{named "John"}x) \to \mathbf{male}x \tag{6.5b}$$
$$\forall x.(\text{named "Mary"}x) \to \mathbf{female}x \tag{6.5c}$$

Assume that the conjunction of formulas (6.5) is abbreviated as `c`. `c` is useful for solving pronominal anaphora in Example 6.19. □

Example 6.4. [Uttering a donkey sentence in a discourse] Recall interpretation (5.66) of the donkey sentence (51), for convenience repeated below with e_{odf} abbreviating $\mathsf{upd}(\mathbf{o}xy, \mathsf{upd}(\mathbf{d}y, \mathsf{upd}(\mathbf{f}x, e)))$:

(51) *Every farmer who owns a donkey beats it.*

[1] In de Groote [2006] the empty context is represented by the empty list `nil` because γ is defined as `list of` $[\iota]$.

$$\mathbf{S}_d = \lambda e\phi.\forall(\lambda x.\mathbf{f}x \to \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \to \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_{odf}))) \wedge \phi e$$

Assume that the sentence is uttered in some discourse. Consider that this discourse has an empty content and its context contains only one formula expressing that whoever/whatever is a donkey is non-human. This discourse can be expressed with the term $(\lambda\phi.\phi\mathsf{c})$, abbreviated below as $\mathbf{D}_0$, where $\mathsf{c} = (\forall x.(\mathbf{d}x \to \mathbf{non_human}x))$. $\mathbf{D}_0$ is updated with $\mathbf{S}_d$ according to (6.3) in the following way:

$$\begin{aligned}
&\mathsf{dupd}\ \mathbf{D}_0\ \mathbf{S}_d \\
&= \lambda\phi.\ \mathbf{D}_0(\lambda e.\mathbf{S}_d\ e\ \phi) \\
&= \lambda\phi.\ (\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathbf{S}_d\ e\ \phi) \\
&\to_\beta \lambda\phi.\ (\lambda e.\mathbf{S}_d\ e\ \phi)\mathsf{c} \\
&\to_\beta \lambda\phi.\mathbf{S}_d\ \mathsf{c}\ \phi \\
&= \lambda\phi.((\lambda e\phi.\forall(\lambda x.\mathbf{f}x \to \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \to \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_{odf}))) \wedge \phi e)\ \mathsf{c}\ \phi) \\
&\to_\beta^* \lambda\phi.\forall(\lambda x.\mathbf{f}x \to \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \to \mathbf{b}x(\mathsf{sel}(\lambda z.\mathbf{non_human}z)e_{odf}))) \wedge\ \phi\ \mathsf{c}
\end{aligned}$$

Since $\mathsf{c} = (\forall x.(\mathbf{d}x \to \mathbf{non_human}x))$, the following holds (taking upd to be the conjunction of arguments):

$$e_{odf} \vdash (\lambda z.\mathbf{non_human}z)y \tag{6.6}$$

Therefore, by Definition 5.4, function sel selects the individual y from the context e_{odf}, resulting in the final interpretation (6.7) of the updated discourse:

$$\lambda\phi.\forall(\lambda x.\mathbf{f}x \to \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \to \mathbf{b}xy)) \wedge \phi\mathsf{c} \tag{6.7}$$

□

In the previous example sel could find the required referent in the provided context. This is, however, not always the case. Recall Definition 5.4 of the selection function sel, for convenience repeated below:

Definition 6.5. Let P be a term of type $(\iota \to o)$ and e be a term of type o.[2] Then sel P e is defined as follows:

$$\mathsf{sel}\ \mathrm{P}\ e = \begin{cases} \mathsf{choose}\{\mathsf{a} \mid e \vdash \mathrm{P}\mathsf{a}\} & \text{if } \{e \vdash \mathrm{P}\mathsf{a}\} \neq \emptyset \\ \mathsf{raise}\ (\texttt{AbsentIndividualExc}\ \mathrm{P}) & \text{otherwise} \end{cases}$$

According to this definition, the selection function can raise an exception (`AbsentIndividualExc` P) when there is no individual satisfying property P in the context e. The context of discourse is provided to a sentence when the discourse is updated with this sentence. Therefore, this exception can be triggered in the term $(\lambda\phi.\mathbf{D}(\lambda e.\mathbf{S}e\phi))$ of Definition 6.1. The exception has to be handled. Consequently, Definition 6.1 can be improved:

[2]Recall that γ is defined as o here.

Definition 6.6. [Discourse Update] Let $\mathbf{D}$ be an interpretation of a discourse and $\mathbf{S}$ be an interpretation of a sentence. Then the function $\mathsf{dupd_g}$ is defined by Equation (6.8a), where gacc is a function of type of type $((\gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o)$ that is responsible for global accommodation and is defined in (6.8b)–(6.8c):

$$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S} \doteq \lambda\phi.\ \mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \tag{6.8a}$$

$$\mathsf{gacc}\ \mathbf{S}\ e\ \phi \doteq \mathbf{S}\ e\ \phi \tag{6.8b}$$

$$\texttt{handle}\ (\texttt{AbsentIndividualExc}\ \mathrm{P})\ \texttt{with}$$
$$\exists(\lambda x.(\mathrm{P}x) \wedge \mathsf{gacc}\ \mathbf{S}\ (\mathsf{upd}(\mathrm{P}x, e))\ \phi) \tag{6.8c}$$

If an exception is raised, it has to be handled and the $\mathsf{dupd_g}$ function above has such a handler for presuppositions triggered by referring expressions with descriptive content P, i.e. a handler of (`AbsentIndividualExc` P). The exception handler performs **accommodation** of the presuppositional content of presuppositions. Thus, the handler of (`AbsentIndividualExc` P) introduces an individual (variable) x into the logical form of the discourse, updates the context with $(\mathrm{P}x)$, and makes a new recursive call of gacc with $\mathbf{S}$ applied to the amended context $(\mathsf{upd}(\mathrm{P}x, e))$ and to the continuation ϕ (note that continuation ϕ is the variable over which the resulting discourse is abstracted). The function gacc is a recursive function because the interpretation $\mathbf{S}$ evaluated on the context e can raise more than one exception.

It is important to see why the exception handling happens on the level of discourse. Each sentence has a dynamic interpretation independent of the discourse it appears in. However, due to the fact that the interpretation of the sentence has an argument standing for the context, it "receives" the context of the discourse in which it is appended. Whether an exception is raised or not depends on the discourse's context and the interpretation of the sentence. That is why the exception handling is performed when the interpretation of an updated discourse is computed.

As a concrete example, consider two miniature discourses in (52). Note that (52a) and (52b) differ only in the first sentence.

(52) a. *The story is about John. John loves Mary.*

b. *The story is about John and Mary. John loves Mary.*

Let $\mathbf{D}_0$ be the initial discourse with an empty context. After $\mathbf{D}_0$ is updated with the first sentence in (52a), $\mathbf{D}_1^a$ is obtained. After $\mathbf{D}_0$ is updated with the first sentence in (52b), $\mathbf{D}_1^b$ is obtained. Even though the initial discourse $\mathbf{D}_0$ is the same in both cases, the interpretations $\mathbf{D}_1^a$ and $\mathbf{D}_1^b$ differ, because the interpretations of the first sentences differ. Particularly, the logical formula and the context of $\mathbf{D}_1^a$ contain content related to *John* and the logical formula and the context of

$\mathbf{D}_1^b$ contain content related to *John* and *Mary*. Then $\mathbf{D}_1^a$ and $\mathbf{D}_1^b$ are updated with the interpretations of the second sentences. Even though the interpretations of the second sentences are identical, the update of $\mathbf{D}_1^a$ with the interpretation of the second sentence causes an exception related to *Mary* but the update of $\mathbf{D}_1^b$ with the interpretation of the second sentence does not cause any exception. However, after the exception during the update of $\mathbf{D}_1^a$ is handled, the resulting interpretations $\mathbf{D}_2^a$ and $\mathbf{D}_2^b$ are logically equivalent and their contexts contain the same propositions.

Summing up, each sentence has its own compositionally computed meaning. The discourse update function serves as an interface between the meanings of the old discourse and of the new sentence; it combines both meanings and results in the meaning of a new, updated, discourse.

Examples 6.7, 6.18 and 6.19 taken together compositionally compute the meaning of (50d).

EXAMPLE 6.7. [$\mathsf{dupd_g}\ \mathbf{D}_0\ \mathbf{S}_1$] Assume that Sentence (50a) is uttered in the discourse $(\lambda\phi.\phi\mathsf{c})$. Taking the interpretation $\mathbf{S}_1$ of Sentence (50a) given in (5.34), the update of $\mathbf{D}_0$ with $\mathbf{S}_1$ is computed in the following way:

$$\begin{aligned}
\mathsf{dupd_g}\ \mathbf{D}_0\ \mathbf{S}_1 &= \lambda\phi.\ \mathbf{D}_0(\lambda e.\mathsf{gacc}\ \mathbf{S}_1\ e\ \phi) && \text{(by (6.8a))}\\
&= \lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}_1\ e\ \phi)\\
&\to_\beta \lambda\phi.(\lambda e.\mathsf{gacc}\ \mathbf{S}_1\ e\ \phi)\mathsf{c}\\
&\to_\beta \lambda\phi.\mathsf{gacc}\ \mathbf{S}_1\ \mathsf{c}\ \phi && (6.9)
\end{aligned}$$

The computation proceeds in the subterm $(\mathsf{gacc}\ \mathbf{S}_1\ \mathsf{c}\ \phi)$. According to the definition of gacc, $(\mathbf{S}_1\ \mathsf{c}\ \phi)$ is computed firstly:

$$\begin{aligned}
&\mathbf{S}_1\ \mathsf{c}\ \phi\\
&= (\lambda e\phi.\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e)\wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e),e)))\mathsf{c}\phi\\
&\to_\beta^* \mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c})(\mathsf{sel}(\text{named "John"})\mathsf{c})\wedge && (6.10)\\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c})(\mathsf{sel}(\text{named "John"})\mathsf{c}),\mathsf{c}))
\end{aligned}$$

Term (6.10) raises two exceptions: in the subterm $(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c})$ and in the subterm $(\mathsf{sel}(\text{named "John"})\mathsf{c})$. These exceptions are raised and handled one after the other. Assume that the first exception to be raised is the exception in the subterm $(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c})$, which expresses the attempt to select an individual having property $(\lambda x.\mathbf{story}x)$ in the context c. The exception is handled by gacc according to Equation (6.8c), leading to (6.11):

$$\begin{aligned}
&\exists(\lambda s.(\lambda x.\mathbf{story}x)s\wedge\mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}((\lambda x.\mathbf{story}x)s,\mathsf{c}))\ \phi)\\
\to_\beta^* &\exists(\lambda s.\mathbf{story}s\wedge\mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\mathbf{story}s,\mathsf{c}))\ \phi) && (6.11)
\end{aligned}$$

Consequently, there is a recursive call to gacc, computing the subterm $(\mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\mathbf{story}s, \mathsf{c}))\ \phi)$. To show the computation more compactly, the term $(\mathsf{upd}(\mathbf{story}s, \mathsf{c}))$ is abbreviated by c_s. By the definition of gacc, $(\mathbf{S}_1\ \mathsf{c}_s\ \phi)$ is computed firstly:

$$\begin{aligned}
&\mathbf{S}_1\ \mathsf{c}_s\ \phi && \text{(by (6.8b))}\\
&= (\lambda e\phi.\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e)\wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e), e)))\mathsf{c}_s\phi\\
&\rightarrow^*_\beta \mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c}_s)(\mathsf{sel}(\text{named "John"})\mathsf{c}_s)\wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c}_s)(\mathsf{sel}(\text{named "John"})\mathsf{c}_s), \mathsf{c}_s))
\end{aligned} \tag{6.12}$$

Since, due to the previous exception handling, the empty context c is updated with $\mathbf{story}s$, there is no problem now in computing $(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c}_s)$ in Equation (6.12) (neither in the logical formula nor in the context). However, there is an exception on property (named "John") during the attempt to select an individual having this property in context c_s, which does not contain such an individual. The exception is handled analogously, according to Equation (6.8c), resulting in (6.13):

$$\exists(\lambda j.(\text{named "John"}\, j) \wedge \mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}_s))\ \phi) \tag{6.13}$$

The computation continues for the subterm $(\mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}_s))\ \phi)$ of the term (6.13). To show the computation in a more compact way, $(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}_s))$ is abbreviated below as c_{sj}. This time an individual with the property $(\lambda x.\mathbf{story}x)$ and an individual with the property (named "John") are successfully selected from the context c_{sj} in Equation (6.14), resulting in Equation (6.15):

$$\begin{aligned}
&\mathbf{S}_1\ \mathsf{c}_{sj}\ \phi && \text{(by (6.8b))}\\
&= (\lambda e\phi.\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e)\wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)e)(\mathsf{sel}(\text{named "John"})e), e)))\mathsf{c}_{sj}\phi\\
&\rightarrow^*_\beta \mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c}_{sj})(\mathsf{sel}(\text{named "John"})\mathsf{c}_{sj})\wedge && (6.14)\\
&\qquad \phi(\mathsf{upd}(\mathbf{about}(\mathsf{sel}(\lambda x.\mathbf{story}x)\mathsf{c}_{sj})(\mathsf{sel}(\text{named "John"})\mathsf{c}_{sj}), \mathsf{c}_{sj}))\\
&= \mathbf{about}sj \wedge \phi(\mathsf{upd}(\mathbf{about}sj, \mathsf{c}_{sj})) && (6.15)
\end{aligned}$$

Combining the β-reductions shown in Equations (6.15),(6.13),(6.11) and (6.9) the result of computing $\mathsf{dupd}_\mathsf{g}\ \mathbf{D}_0\ \mathbf{S}_1$ is the following:

$$\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}\, j) \wedge (\mathbf{about}sj \wedge \phi(\mathsf{upd}(\mathbf{about}sj, \mathsf{c}_{sj})))))$$

After unfolding c_{sj}, the following term is obtained:

$$\begin{aligned}\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}\, j) \wedge (\mathbf{about}sj \wedge \\ \phi(\mathsf{upd}(\mathbf{about}sj, \\ \mathsf{upd}(\text{named "John"}\, j, \\ \mathsf{upd}(\mathbf{story}s, \mathsf{c}))))))\end{aligned} \tag{6.16}$$

□

In Example 6.7 the first exception raised and handled is the exception related to *the story*, and the second exception is the exception related to *John*. However, the order of exception raising and handling within the same discourse update does not play a role, since the resulting formulas representing the meaning of the updated discourse are logically equivalent.

6.2 Framework GLχ

The addition of exception raising and handling requires the extension of the lambda calculus defined in Section 4.3.

Definition 6.8. [λ-terms] The set of **λ-terms** T is defined as follows:

$$T ::= x \mid k \mid (ET) \mid \lambda x.T \mid (TT) \mid (\mathsf{raise}\ T) \mid T\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ T$$

where x is a variable, k is a constant, E is an exception constructor.

In the following definition an additional type χ for exceptions is added:

Definition 6.9. [Types] The set T of types of framework GLχ is defined inductively as follows:

$$\begin{array}{rll} \text{Atomic types:} & \iota \in \mathrm{T} & \text{(individual)} \\ & o \in \mathrm{T} & \text{(proposition)} \\ & \gamma \in \mathrm{T} & \text{(context)} \\ & \chi \in \mathrm{T} & \text{(exception)} \\ \\ \text{Complex types:} & \alpha, \beta \in \mathrm{T} \Longrightarrow & (\alpha \to \beta) \in \mathrm{T} \end{array}$$

The typing rules are typing rules given in Definition 4.25 plus typing rules defined in 6.10:

Definition 6.10. [Typing rules] Let α, β be arbitrary types and E be an exception constructor of type $(\beta \to \chi)$. A statement $t : \delta$ is **derivable** from the basis Δ, i.e. $\Delta \vdash t : \delta$, if $t : \delta$ can be produced using rules defined in 4.25 and rules given below:

$$\frac{}{\Gamma \vdash E : \beta \to \chi}$$

$$\frac{\Gamma \vdash e : \chi}{\Gamma \vdash \mathsf{raise}\ e : \alpha}$$

$$\frac{\Gamma \vdash t_1 : \alpha \qquad \Gamma \vdash E : \beta \to \chi \qquad \Gamma, t_2 : \beta \vdash t_3 : \alpha}{\Gamma \vdash t_1\ \mathtt{handle}\ (Et_2)\ \mathtt{with}\ t_3 : \alpha}$$

Definition 6.11. [Strong and weak normal forms] Let x and y be variables, k be a constant, E be an exception constructor. Terms in **strong** (S) **normal form** and in **weak** (W) **normal form** are defined as follows:

$$\begin{aligned} Q &::= x \mid k \mid (ES) \mid (QS) \\ S &::= Q \mid \lambda y.\mathsf{raise}\ S[y] \mid \lambda y.S[y] \\ W &::= Q \mid \lambda x.T \end{aligned}$$

Definition 6.12. [Uncaught exception] **Uncaught exceptions** are defined as follows:

$$U ::= \mathsf{raise}\ S$$

Definition 6.13. [Weak and strong evaluation rules] Let x be a variable, k be a constant, E, E_1 and E_2 be exception constructors, t, t_1, t_2 and t_3 be terms, w, w_1 and w_2 range over weak normal forms, v range over strong normal forms, u range over strong normal forms and uncaught exceptions. Terms in framework GLχ are evaluated according to the following rules:

Weak evaluation rules:

$$\frac{}{x \to_w x}$$

$$\frac{}{k \to_w k}$$

$$\frac{}{\lambda x.t \to_w \lambda x.t}$$

$$\frac{t \to_s v}{Et \to_w Ev}$$

$$\frac{t \to_s \mathsf{raise}\ v}{Et \to_w \mathsf{raise}\ v}$$

$$\frac{t_1 \to_w \lambda x.t_3 \qquad t_2 \to_w w_1 \qquad t_3[x := w_1] \to_w w_2}{t_1 t_2 \to_w w_2}$$

$$\frac{t_1 \to_w \lambda x.t_3 \qquad t_2 \to_w \mathsf{raise}\ v}{t_1 t_2 \to_w \mathsf{raise}\ v}$$

$$\frac{t_1 \to_w \mathsf{raise}\ v}{t_1 t_2 \to_w \mathsf{raise}\ v}$$

$$\frac{t_1 \to_w w \qquad t_2 \to_s \mathsf{raise}\ v}{t_1 t_2 \to_w \mathsf{raise}\ v}\ w \text{ is not a } \lambda\text{-abstraction}$$

$$\frac{t_1 \to_w w \qquad t_2 \to_s v}{t_1 t_2 \to_w wv}\ w \text{ is not a } \lambda\text{-abstraction}$$

$$\frac{t \to_s v}{\mathsf{raise}\ t \to_w \mathsf{raise}\ v}$$

$$\frac{t \to_s \mathsf{raise}\ v}{\mathsf{raise}\ t \to_w \mathsf{raise}\ v}$$

$$\frac{t_1 \to_s \mathsf{raise}\ (Ev) \qquad t_2 \to_w w_1 \qquad w_1[x := v] \to_w w_2}{t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 \to_w w_2}$$

$$\frac{t_1 \to_s \mathsf{raise}\ (E_2 v)}{t_1\ \mathtt{handle}\ (E_1 x)\ \mathtt{with}\ t_2 \to_w \mathsf{raise}\ (E_2 v)}\ E_1 \neq E_2$$

$$\frac{t_1 \to_s v}{t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 \to_w v}$$

Strong evaluation rules:

$$\frac{t \to_w \lambda x.t_1 \qquad t_1 \to_s v}{t \to_s \lambda x.v}$$

$$\frac{t \to_w \lambda x.t_1 \qquad t_1 \to_s \mathsf{raise}\ v}{t \to_s \mathsf{raise}\ v}\ \text{provided } x \notin FV(v)$$

$$\frac{t \to_w \lambda x.t_1 \qquad t_1 \to_s \mathsf{raise}\ v}{t \to_s \lambda x.\mathsf{raise}\ v}\ \text{provided } x \in FV(v)$$

$$\frac{t \to_w u}{t \to_s u}$$

The idea of the evaluation system defined above is as follows. The normal form of a term should be in $S \cup U$. The evaluation of a term starts by weakly evaluating it. Weak evaluation directly results in a term in $S \cup U$ (due to the last strong evaluation rule) except when the evaluated term is an abstraction (in this case the evaluation proceeds according to one of the first three strong evaluation rules).
Subsequently, the following notation is used for convenience:

NOTATION 6.14. Evaluation $\rightarrow_{eval}$ abbreviates $\rightarrow_w \cup \rightarrow_s$.

The next proposition proves that the rules in Definition 6.13 always yield either a normal form or an uncaught exception. Moreover, for each term to be evaluated, Definition 6.13 provides a single rule that can be applied. This guarantees the uniqueness of term evaluation in the framework.

PROPOSITION 6.15.

1. *If $t \rightarrow_s t'$, then $t' \in S \cup U$.*
2. *If $t \rightarrow_w t'$, then $t' \in W \cup U$.*

Proof. The proof is by induction on the derivation φ of $t \rightarrow_{eval} t'$.

- If φ is of the form

$$\frac{}{x \rightarrow_w x}$$

  then $x \in W$ by Definition 6.11.

- If φ is of the form

$$\frac{}{k \rightarrow_w k}$$

  then $k \in W$ by Definition 6.11.

- If φ is of the form

$$\frac{}{\lambda x.t \rightarrow_w \lambda x.t}$$

  then $\lambda x.t \in W$ by Definition 6.11.

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t \to_s v\end{array}}{Et \to_w Ev}$$

then, by I.H., $v \in S \cup U$. By Definition 6.13, v ranges over strong normal forms, hence $v \in S$. Therefore, $Ev \in W$ by Definition 6.11.

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t \to_s \mathsf{raise}\ v\end{array}}{Et \to_w \mathsf{raise}\ v}$$

then, by I.H., $\mathsf{raise}\ v \in S \cup U$. However, by Definition 6.12, However, $\mathsf{raise}\ v$ cannot be in S because it starts with raise. Therefore, $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\end{array} \quad \begin{array}{c}\varphi''\end{array} \quad \begin{array}{c}\varphi'''\end{array}}{\begin{array}{ccc}t_1 \to_w \lambda x.t_3 & t_2 \to_w w_1 & t_3[x := w_1] \to_w w_2\end{array}}$$
$$t_1 t_2 \to_w w_2$$

then, by I.H., $w_2 \in W \cup U$.

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\end{array} \quad \begin{array}{c}\varphi''\end{array}}{\begin{array}{cc}t_1 \to_w \lambda x.t_3 & t_2 \to_w \mathsf{raise}\ v\end{array}}$$
$$t_1 t_2 \to_w \mathsf{raise}\ v$$

then, by I.H., $\mathsf{raise}\ v \in W \cup U$. However, $\mathsf{raise}\ v$ cannot be in W as it starts with raise. Therefore, $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t_1 \to_w \mathsf{raise}\ v\end{array}}{t_1 t_2 \to_w \mathsf{raise}\ v}$$

then, by I.H., $\mathsf{raise}\ v \in W \cup U$. However, $\mathsf{raise}\ v$ cannot be in W as it starts with raise. Therefore, $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t_1 \to_w w} \qquad \overset{\varphi''}{t_2 \to_s \mathsf{raise}\ v}}{t_1 t_2 \to_w \mathsf{raise}\ v}\ w \text{ is not a } \lambda\text{-abstraction}$$

then, by I.H., $\mathsf{raise}\ v \in S \cup U$. However, $\mathsf{raise}\ v$ cannot be in S as it starts with raise. Therefore, $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t_1 \to_w w} \qquad \overset{\varphi''}{t_2 \to_s v}}{t_1 t_2 \to_w wv}\ w \text{ is not a } \lambda\text{-abstraction}$$

then, by I.H., $w \in W \cup U$ and $v \in S \cup U$. By Definition 6.13, w ranges over weak normal forms and v ranges over strong normal forms, i.e. $w \in W$ and $v \in S$. Since w is not a λ-abstraction, $w \in Q$. Then, by Definition 6.11, $wv \in Q$, and, therefore, $wv \in W$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t \to_s v}}{\mathsf{raise}\ t \to_w \mathsf{raise}\ v}$$

then, by I.H., $v \in S \cup U$. By Definition 6.13 v, ranges over strong normal forms, i.e. $v \in S$. Then $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t \to_s \mathsf{raise}\ v}}{\mathsf{raise}\ t \to_w \mathsf{raise}\ v}$$

then, by I.H., $\mathsf{raise}\ v \in S \cup U$. $\mathsf{raise}\ v$ cannot be in S as it starts with raise. Therefore, $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t_1 \to_s \mathsf{raise}\ (Ev)} \qquad \overset{\varphi''}{t_2 \to_w w_1} \qquad \overset{\varphi'''}{w_1[x := v] \to_w w_2}}{t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 \to_w w_2}$$

then, by I.H., $w_2 \in W \cup U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t_1 \to_s \mathsf{raise}\ (E_2 v)}}{t_1\ \mathtt{handle}\ (E_1 x)\ \mathtt{with}\ t_2 \to_w \mathsf{raise}\ (E_2 v)}\ E_1 \neq E_2$$

then, by I.H., $\mathsf{raise}\ (E_2 v) \in S \cup U$. However, $\mathsf{raise}\ (E_2 v)$ cannot be in S as it starts with raise. Therefore, $\mathsf{raise}\ (E_2 v) \in U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t_1 \to_s v}}{t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 \to_w v}$$

then, by I.H., $v \in S \cup U$. Hence, $v \in W \cup U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t \to_w \lambda x.t_1} \qquad \overset{\varphi''}{t_1 \to_s v}}{t \to_s \lambda x.v}$$

then, by I.H. $v \in S \cup U$. By Definition 6.13 v ranges over strong normal forms, hence $v \in S$. Then, by Definition 6.11, $\lambda x.v \in S$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t \to_w \lambda x.t_1} \qquad \overset{\varphi''}{t_1 \to_s \mathsf{raise}\ v}}{t \to_s \mathsf{raise}\ v}\ \text{provided } x \notin FV(v)$$

then, by I.H., $\mathsf{raise}\ v \in S \cup U$. However, $\mathsf{raise}\ v$ cannot be in S as it starts with raise. Therefore, $\mathsf{raise}\ v \in U$.

- If φ is of the form

$$\frac{\overset{\varphi'}{t \to_w \lambda x.t_1} \qquad \overset{\varphi''}{t_1 \to_s \mathsf{raise}\ v}}{t \to_s \lambda x.\mathsf{raise}\ v}\ \text{provided } x \in FV(v)$$

then, by I.H., $\mathsf{raise}\ v \in S \cup U$. However, $\mathsf{raise}\ v$ cannot be in S as it starts with raise. Therefore, $\mathsf{raise}\ v \in U$. Then, by Definition 6.12, v must be in S. Hence, from $x \in FV(v)$ it follows by Definition 6.11 that $\lambda x.\mathsf{raise}\ v \in S$.

- When φ is of the form

$$\dfrac{\dfrac{\overset{\varphi'}{\overline{t \to_w u}}}{t \to_s u}}$$

Then, $u \in S \cup U$ by Definition 6.13.

□

Proposition 6.16. [Subject Reduction]

1. *if* $t \to_w t'$ *and* $\Gamma \vdash t : \delta$ *then* $\Gamma \vdash t' : \delta$
2. *if* $t \to_s t'$ *and* $\Gamma \vdash t : \delta$ *then* $\Gamma \vdash t' : \delta$

Proof. The proof is by induction on the derivation φ.

- If φ is of the form

$$\dfrac{}{x \to_w x}$$

and $\Gamma \vdash x : \delta$, then trivially $\Gamma \vdash x : \delta$.

- If φ is of the form

$$\dfrac{}{k \to_w k}$$

and $\Gamma \vdash k : \delta$, then trivially $\Gamma \vdash k : \delta$.

- If φ is of the form

$$\dfrac{}{\lambda x.t \to_w \lambda x.t}$$

and $\Gamma \vdash \lambda x.t : \alpha \to \beta$, then trivially $\Gamma \vdash \lambda x.t : \alpha \to \beta$.

- If φ is of the form

$$\dfrac{\dfrac{\overset{\varphi'}{\overline{t \to_s v}}}{Et \to_w Ev}}$$

and $\Gamma \vdash Et : \chi$, we need to prove $\Gamma \vdash Ev : \chi$. Any derivation of the sequent $\Gamma \vdash Et : \chi$ is of the following form:

$$\frac{\begin{array}{c}\psi'\\ \Gamma \vdash E : \alpha \to \chi\end{array} \qquad \begin{array}{c}\psi''\\ \Gamma \vdash t : \alpha\end{array}}{\Gamma \vdash Et : \chi}$$

By I.H., it follows from $t \to_s v$ and $\Gamma \vdash t : \alpha$ that there exists a derivation ψ''' of $\Gamma \vdash v : \alpha$. Then, the derivation of $\Gamma \vdash Ev : \chi$ can be constructed as follows:

$$\frac{\begin{array}{c}\psi'\\ \Gamma \vdash E : \alpha \to \chi\end{array} \qquad \begin{array}{c}\psi'''\\ \Gamma \vdash v : \alpha\end{array}}{\Gamma \vdash Ev : \chi}$$

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t \to_s \mathsf{raise}\ v\end{array}}{Et \to_w \mathsf{raise}\ v}$$

and $\Gamma \vdash Et : \chi$, we need to prove $\Gamma \vdash \mathsf{raise}\ v : \chi$. Any derivation of the sequent $\Gamma \vdash Et : \chi$ is of the following form:

$$\frac{\begin{array}{c}\psi'\\ \Gamma \vdash E : \alpha \to \chi\end{array} \qquad \begin{array}{c}\psi''\\ \Gamma \vdash t : \alpha\end{array}}{\Gamma \vdash Et : \chi}$$

From $\Gamma \vdash t : \alpha$ and $t \to_s \mathsf{raise}\ v$ it follows, by I.H., that there exists a derivation ψ of $\Gamma \vdash \mathsf{raise}\ v : \alpha$ necessarily of the following form:

$$\frac{\begin{array}{c}\psi'''\\ \Gamma \vdash v : \chi\end{array}}{\Gamma \vdash \mathsf{raise}\ v : \alpha}$$

Then, the derivation of $\Gamma \vdash \mathsf{raise}\ v : \chi$ can be constructed as follows:

$$\frac{\begin{array}{c}\psi'''\\ \Gamma \vdash v : \chi\end{array}}{\Gamma \vdash \mathsf{raise}\ v : \chi}$$

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t_1 \to_w \lambda x.t_3\end{array} \qquad \begin{array}{c}\varphi''\\ t_2 \to_w w_1\end{array} \qquad \begin{array}{c}\varphi'''\\ t_3[x := w_1] \to_w w_2\end{array}}{t_1 t_2 \to_w w_2}$$

and $\Gamma \vdash t_1 t_2 : \beta$, we need to prove that $\Gamma \vdash w_2 : \beta$. Any derivation of the sequent $\Gamma \vdash t_1 t_2 : \beta$ is of the following form:

$$\dfrac{\dfrac{\overset{\psi'}{\vdots}}{\Gamma \vdash t_1 : \alpha \to \beta} \qquad \dfrac{\overset{\psi''}{\vdots}}{\Gamma \vdash t_2 : \alpha}}{\Gamma \vdash t_1 t_2 : \beta}$$

From $\Gamma \vdash t_1 : \alpha \to \beta$ and $t_1 \to_w \lambda x.t_3$ it follows, by I.H., that $\Gamma \vdash \lambda x.t_3 : \alpha \to \beta$. From $\Gamma \vdash t_2 : \alpha$ and $t_2 \to_w w_1$ it follows, by I.H., that $\Gamma \vdash w_1 : \alpha$. By the substitution lemma A.28, $\Gamma \vdash t_3[x := w_3] : \beta$. Then, by I.H. $\Gamma \vdash w_2 : \beta$.

- If φ is of the form

$$\dfrac{\dfrac{\overset{\varphi'}{\vdots}}{t_1 \to_w \lambda x.t_3} \qquad \dfrac{\overset{\varphi''}{\vdots}}{t_2 \to_w \mathsf{raise}\ v}}{t_1 t_2 \to_w \mathsf{raise}\ v}$$

and $\Gamma \vdash t_1 t_2 : \beta$, we need to prove that $\Gamma \vdash \mathsf{raise}\ v : \beta$. Any derivation of the sequent $\Gamma \vdash t_1 t_2 : \beta$ is of the following form:

$$\dfrac{\dfrac{\overset{\psi'}{\vdots}}{\Gamma \vdash t_1 : \alpha \to \beta} \qquad \dfrac{\overset{\psi''}{\vdots}}{\Gamma \vdash t_2 : \alpha}}{\Gamma \vdash t_1 t_2 : \beta}$$

From $\Gamma \vdash t_2 : \alpha$ and $t_2 \to_w \mathsf{raise}\ v$ it follows, by I.H., that there exists a derivation ψ of $\Gamma \vdash \mathsf{raise}\ v : \alpha$ necessarily of the following form:

$$\dfrac{\overset{\psi'''}{\vdots}}{\Gamma \vdash v : \chi}{\Gamma \vdash \mathsf{raise}\ v : \alpha}$$

Then, the derivation of $\Gamma \vdash \mathsf{raise}\ v : \beta$ can be constructed as follows:

$$\dfrac{\overset{\psi'''}{\vdots}}{\Gamma \vdash v : \chi}{\Gamma \vdash \mathsf{raise}\ v : \beta}$$

- If φ is of the form

$$\dfrac{\overset{\varphi'}{\vdots}}{t_1 \to_w \mathsf{raise}\ v}{t_1 t_2 \to_w \mathsf{raise}\ v}$$

and $\Gamma \vdash t_1 t_2 : \beta$, we need to prove that $\Gamma \vdash \mathsf{raise}\ v : \beta$. Any derivation of the sequent $\Gamma \vdash t_1 t_2 : \beta$ is of the following form:

$$\frac{\overset{\psi'}{\Gamma \vdash t_1 : \alpha \to \beta} \qquad \overset{\psi''}{\Gamma \vdash t_2 : \alpha}}{\Gamma \vdash t_1 t_2 : \beta}$$

From $\Gamma \vdash t_1 : \alpha \to \beta$ and $t_1 \to_w \mathsf{raise}\ v$ it follows, by I.H., that there exists a derivation ψ of $\Gamma \vdash \mathsf{raise}\ v : \alpha \to \beta$. ψ must be of the following form:

$$\frac{\overset{\psi''}{\Gamma \vdash v : \chi}}{\Gamma \vdash \mathsf{raise}\ v : \alpha \to \beta}$$

Then, the derivation of $\Gamma \vdash \mathsf{raise}\ v : \beta$ can be constructed as follows:

$$\frac{\overset{\psi''}{\Gamma \vdash v : \chi}}{\Gamma \vdash \mathsf{raise}\ v : \beta}$$

- If φ is of the form

$$\frac{\overset{\varphi'}{t_1 \to_w w} \qquad \overset{\varphi''}{t_2 \to_s \mathsf{raise}\ v}}{t_1 t_2 \to_w \mathsf{raise}\ v}\ w \text{ is not a } \lambda\text{-abstraction}$$

and $\Gamma \vdash t_1 t_2 : \beta$, we need to prove that $\Gamma \vdash \mathsf{raise}\ v : \beta$. Any derivation of the sequent $\Gamma \vdash t_1 t_2 : \beta$ is of the following form:

$$\frac{\overset{\psi'}{\Gamma \vdash t_1 : \alpha \to \beta} \qquad \overset{\psi''}{\Gamma \vdash t_2 : \alpha}}{\Gamma \vdash t_1 t_2 : \beta}$$

From $\Gamma \vdash t_2 : \alpha$ and $t_2 \to_s \mathsf{raise}\ v$ it follows, by I.H., that there exists a derivation ψ of $\Gamma \vdash \mathsf{raise}\ v : \alpha$. ψ is of the following form:

$$\frac{\overset{\psi''}{\Gamma \vdash v : \chi}}{\Gamma \vdash \mathsf{raise}\ v : \alpha}$$

Then, the derivation of $\Gamma \vdash \mathsf{raise}\ v : \beta$ can be constructed as follows:

$$\dfrac{\dfrac{\overset{\psi''}{\vdots}}{\Gamma \vdash v : \chi}}{\Gamma \vdash \mathsf{raise}\ v : \beta}$$

- If φ is of the form

$$\dfrac{\dfrac{\overset{\varphi'}{\vdots}}{t_1 \to_w w} \qquad \dfrac{\overset{\varphi''}{\vdots}}{t_2 \to_s v}}{t_1 t_2 \to_w wv}\ w \text{ is not a } \lambda\text{-abstraction}$$

and $\Gamma \vdash t_1 t_2 : \beta$, we need to prove $\Gamma \vdash wv : \beta$. Any derivation of the sequent $\Gamma \vdash t_1 t_2 : \beta$ is of the following form:

$$\dfrac{\dfrac{\overset{\psi'}{\vdots}}{\Gamma \vdash t_1 : \alpha \to \beta} \qquad \dfrac{\overset{\psi''}{\vdots}}{\Gamma \vdash t_2 : \alpha}}{\Gamma \vdash t_1 t_2 : \beta}$$

From $t_1 \to_w w$ and $\Gamma \vdash t_1 : \alpha \to \beta$ it follows, by I.H., that there exists a derivation ψ''' of $\Gamma \vdash w : \alpha \to \beta$. From $t_2 \to_s v$ and $\Gamma \vdash t_2 : \alpha$ it follows, by I.H., that that there exists a derivation ψ'''' of $\Gamma \vdash v : \alpha$. Then, the derivation of $\Gamma \vdash wv : \beta$ can be constructed as follows:

$$\dfrac{\dfrac{\overset{\psi'''}{\vdots}}{\Gamma \vdash w : \alpha \to \beta} \qquad \dfrac{\overset{\psi''''}{\vdots}}{\Gamma \vdash v : \alpha}}{\Gamma \vdash wv : \beta}$$

- If φ is of the form

$$\dfrac{\dfrac{\overset{\varphi'}{\vdots}}{t \to_s v}}{\mathsf{raise}\ t \to_w \mathsf{raise}\ v}$$

and $\Gamma \vdash \mathsf{raise}\ t : \alpha$, then we need to prove $\Gamma \vdash \mathsf{raise}\ v : \alpha$. Any derivation of the sequent $\Gamma \vdash \mathsf{raise}\ t : \alpha$ is of the following form:

$$\dfrac{\dfrac{\overset{\psi'}{\vdots}}{\Gamma \vdash t : \chi}}{\Gamma \vdash \mathsf{raise}\ t : \alpha}$$

Then, from $\Gamma \vdash t : \chi$ and $t \to_s v$ it follows, by I.H., that there is a derivation ψ'' of $\Gamma \vdash v : \chi$. Then, the derivation of $\Gamma \vdash \mathsf{raise}\ v : \alpha$ can be constructed as follows:

$$\frac{\begin{array}{c}\psi''\\ \Gamma \vdash v : \chi\end{array}}{\Gamma \vdash \mathsf{raise}\ v : \alpha}$$

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t \to_s \mathsf{raise}\ v\end{array}}{\mathsf{raise}\ t \to_w \mathsf{raise}\ v}$$

and $\Gamma \vdash \mathsf{raise}\ t : \alpha$, then we need to prove $\Gamma \vdash \mathsf{raise}\ v : \alpha$. Any derivation of the sequent $\Gamma \vdash \mathsf{raise}\ t : \alpha$ is of the following form:

$$\frac{\begin{array}{c}\psi'\\ \Gamma \vdash t : \chi\end{array}}{\Gamma \vdash \mathsf{raise}\ t : \alpha}$$

Then, from $\Gamma \vdash t : \chi$ and $t \to_s \mathsf{raise}\ v$ it follows, by I.H., that there is a derivation ψ of $\Gamma \vdash \mathsf{raise}\ v : \chi$ necessarily of the following form:

$$\frac{\begin{array}{c}\psi''\\ \Gamma \vdash v : \chi\end{array}}{\Gamma \vdash \mathsf{raise}\ v : \chi}$$

Then the derivation of $\Gamma \vdash \mathsf{raise}\ v : \alpha$ can be constructed as follows:

$$\frac{\begin{array}{c}\psi''\\ \Gamma \vdash v : \chi\end{array}}{\Gamma \vdash \mathsf{raise}\ v : \alpha}$$

- If φ is of the form

$$\frac{\begin{array}{c}\varphi'\\ t_1 \to_s \mathsf{raise}\ (Ev)\end{array} \qquad \begin{array}{c}\varphi''\\ t_2 \to_w w_1\end{array} \qquad \begin{array}{c}\varphi'''\\ w_1[x := v] \to_w w_2\end{array}}{t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 \to_w w_2}$$

and $\Gamma \vdash t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 : \alpha$, we need to prove $\Gamma \vdash w_2 : \alpha$. Any derivation of the sequent $\Gamma \vdash t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 : \alpha$ is of the following form:

$$\frac{\begin{array}{c}\psi'\\ \Gamma \vdash t_1 : \alpha\end{array} \qquad \begin{array}{c}\psi''\\ \Gamma \vdash E : \beta \to \chi\end{array} \qquad \begin{array}{c}\psi'''\\ \Gamma, x : \beta \vdash t_2 : \alpha\end{array}}{\Gamma \vdash t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 : \alpha}$$

From $t_2 \to_w w_1$ and $\Gamma, x : \beta \vdash t_2 : \alpha$ it follows, by I.H., that $\Gamma, x : \beta \vdash w_1 : \alpha$. From $t_1 \to_s \mathsf{raise}\ (Ev)$ and $\Gamma \vdash t_1 : \alpha$ it follows, by I.H., that there exists the derivation ψ of $\Gamma \vdash \mathsf{raise}\ (Ev) : \alpha$ of the following form:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash E : \beta \to \chi} \quad \dfrac{\overset{\psi''''}{\vdots}}{\Gamma \vdash v : \beta}}{\Gamma \vdash Ev : \chi}}{\Gamma \vdash \mathsf{raise}\ (Ev) : \alpha}$$

From $\Gamma \vdash v : \beta$ and $\Gamma, x : \beta \vdash t_2 : \alpha$ it follows by the substitution lemma A.28 that $\Gamma \vdash w_1[x := v] : \alpha$. Then from $\Gamma \vdash w_1[x := v] : \alpha$ and $w_1[x := v] \to_w w_2$ it follows, by I.H, that $\Gamma \vdash w_2 : \alpha$.

- If φ is of the form

$$\dfrac{\overset{\varphi'}{\vdots}}{t_1 \to_s \mathsf{raise}\ (E_2 v)}\ E_1 \neq E_2$$

$$\dfrac{}{t_1\ \mathtt{handle}\ (E_1 x)\ \mathtt{with}\ t_2 \to_w \mathsf{raise}\ (E_2 v)}$$

and $\Gamma \vdash t_1\ \mathtt{handle}\ (E_1 x)\ \mathtt{with}\ t_2 : \alpha$, then we need to prove that $\Gamma \vdash \mathsf{raise}\ (E_2 v) : \alpha$. Any derivation of the sequent $\Gamma \vdash t_1\ \mathtt{handle}\ (E_1 x)\ \mathtt{with}\ t_2 : \alpha$ is of the following form:

$$\dfrac{\overset{\psi'}{\vdots} \quad \overset{\psi''}{\vdots} \quad \overset{\psi'''}{\vdots}}{\Gamma \vdash t_1 : \alpha \quad \Gamma \vdash E_1 : \beta \to \chi \quad \Gamma, x : \beta \vdash t_2 : \alpha}$$

$$\Gamma \vdash t_1\ \mathtt{handle}\ (E_1 x)\ \mathtt{with}\ t_2 : \alpha$$

Then, from $\Gamma \vdash t_1 : \alpha$ and $t_1 \to_s \mathsf{raise}\ (E_2 v)$ it follows, by I.H., that $\Gamma \vdash \mathsf{raise}\ (E_2 v) : \alpha$.

- If φ is of the form

$$\dfrac{\overset{\varphi'}{\vdots}}{t_1 \to_s v}$$

$$\dfrac{}{t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 \to_w v}$$

and $\Gamma \vdash t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 : \alpha$, then we need to prove that $\Gamma \vdash v : \alpha$. Any derivation of the sequent $\Gamma \vdash t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 : \alpha$ is of the following form:

$$\dfrac{\overset{\psi'}{\vdots} \quad \overset{\psi''}{\vdots} \quad \overset{\psi'''}{\vdots}}{\Gamma \vdash t_1 : \alpha \quad \Gamma \vdash E : \beta \to \chi \quad \Gamma, x : \beta \vdash t_2 : \alpha}$$

$$\Gamma \vdash t_1\ \mathtt{handle}\ (Ex)\ \mathtt{with}\ t_2 : \alpha$$

Then, from $\Gamma \vdash t_1 : \alpha$ and $t_1 \to_s v$ it follows, by I.H., that $\Gamma \vdash v : \alpha$.

- If φ is of the form

$$\frac{\begin{array}{cc}\overset{\varphi'}{\vdots} & \overset{\varphi''}{\vdots} \\ t \to_w \lambda x.t_1 & t_1 \to_s v\end{array}}{t \to_s \lambda x.v}$$

and $\Gamma \vdash t : \alpha$, then we need to prove that $\Gamma \vdash \lambda x.v : \alpha$. From $\Gamma \vdash t : \alpha$ and $t \to_w \lambda x.t_1$ it follows, by I.H., that α is of the form $(\alpha_1 \to \alpha_2)$ and that there is a derivation $\Gamma \vdash \lambda x.t_1 : \alpha_1 \to \alpha_2$ of the following form:

$$\frac{\overset{\psi'}{\vdots}}{\Gamma, x : \alpha_1 \vdash t_1 : \alpha_2}$$
$$\Gamma \vdash \lambda x.t_1 : \alpha_1 \to \alpha_2$$

From $\Gamma, x : \alpha_1 \vdash t_1 : \alpha_2$ and $t_1 \to_s v$ it follows, by I.H., that there exists a derivation ψ'' of $\Gamma, x : \alpha_1 \vdash v : \alpha_2$. Then, $\Gamma \vdash \lambda x.v : \alpha_1 \to \alpha_2$ can be derived by using the abstraction rule:

$$\frac{\overset{\psi''}{\vdots}}{\Gamma, x : \alpha_1 \vdash v : \alpha_2}$$
$$\Gamma \vdash \lambda x.v : \alpha_1 \to \alpha_2$$

- If φ is of the form

$$\frac{\begin{array}{cc}\overset{\varphi'}{\vdots} & \overset{\varphi''}{\vdots} \\ t \to_w \lambda x.t_1 & t_1 \to_s \mathsf{raise}\ v\end{array}}{t \to_s \mathsf{raise}\ v} \text{ provided } x \notin FV(v)$$

and $\Gamma \vdash t : \alpha$, then we need to prove that $\Gamma \vdash \mathsf{raise}\ v : \alpha$. From $\Gamma \vdash t : \alpha$ and $t \to_w \lambda x.t_1$ it follows, by I.H., that $\Gamma \vdash \lambda x.t_1 : \alpha$. Hence, α is of the form $(\alpha_1 \to \alpha_2)$. Any derivation of the sequent $\Gamma \vdash \lambda x.t_1 : \alpha$ is of the following form:

$$\frac{\overset{\psi'}{\vdots}}{\Gamma, x : \alpha_1 \vdash t_1 : \alpha_2}$$
$$\Gamma \vdash \lambda x.t_1 : \alpha_1 \to \alpha_2$$

Then from $\Gamma, x : \alpha_1 \vdash t_1 : \alpha_2$ and $t_1 \to_s \mathsf{raise}\ v$ it follows, by I.H., that $\Gamma, x : \alpha_1 \vdash \mathsf{raise}\ v : \alpha_2$. Therefore, since $x \notin FV(v)$, the derivation of $\Gamma \vdash \mathsf{raise}\ v : \alpha_2$ can be constructed as follows:

$$\dfrac{\dfrac{\overset{\psi''}{\vdots}}{\Gamma \vdash v : \chi}}{\Gamma \vdash \mathsf{raise}\ v : \alpha_2}$$

Then the derivation of $\Gamma \vdash \mathsf{raise}\ v : \alpha$ can be constructed as follows:

$$\dfrac{\dfrac{\overset{\psi''}{\vdots}}{\Gamma \vdash v : \chi}}{\Gamma \vdash \mathsf{raise}\ v : \alpha}$$

- If φ is of the form

$$\dfrac{\overset{\varphi'}{\vdots} \quad \overset{\varphi''}{\vdots}}{t \to_s \lambda x.\mathsf{raise}\ v} \begin{array}{l} t \to_w \lambda x.t_1 \quad t_1 \to_s \mathsf{raise}\ v \\ \text{provided } x \in FV(v) \end{array}$$

and $\Gamma \vdash t : \alpha$, then we need to prove that $\Gamma \vdash \lambda x.\mathsf{raise}\ v : \alpha$. From $\Gamma \vdash t : \alpha$ and $t \to_w \lambda x.t_1$ it follows, by I.H., that $\Gamma \vdash \lambda x.t_1 : \alpha$. Any derivation of the sequent $\Gamma \vdash \lambda x.t_1 : \alpha$ is of the following form:

$$\dfrac{\dfrac{\overset{\psi'}{\vdots}}{\Gamma, x : \alpha_1 \vdash t_1 : \alpha_2}}{\Gamma \vdash \lambda x.t_1 : \alpha}$$

Hence, α is of the form $(\alpha_1 \to \alpha_2)$. Then, from $\Gamma, x : \alpha_1 \vdash t_1 : \alpha_2$ and $t_1 \to_s \mathsf{raise}\ v$ it follows, by I.H., that there exists a derivation ψ'' of $\Gamma, x : \alpha_1 \vdash \mathsf{raise}\ v : \alpha_2$. Then, the derivation of $\Gamma \vdash \lambda x.\mathsf{raise}\ v : \alpha$ can be constructed as follows:

$$\dfrac{\dfrac{\overset{\psi''}{\vdots}}{\Gamma, x : \alpha_1 \vdash \mathsf{raise}\ v : \alpha_2}}{\Gamma \vdash \lambda x.\mathsf{raise}\ v : \alpha_1 \to \alpha_2}$$

- If φ is of the form

$$\dfrac{\dfrac{\overset{\varphi'}{\vdots}}{t \to_w u}}{t \to_s u}$$

and $\Gamma \vdash t : \alpha$, then we need to prove that $\Gamma \vdash u : \alpha$. From $\Gamma \vdash t : \alpha$ and $t \to_w u$ it follows, by I.H., that $\Gamma \vdash u : \alpha$.

□

Assume discourse $(\lambda\phi.\phi\mathsf{c})$ with context c is updated with sentence $\mathbf{S}$. Then $(\mathsf{dupd}\ (\lambda\phi'.\phi'\mathsf{c})\ \mathbf{S})$ is, by Definition 6.6, equal to $(\lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi))$. The next example shows that the term $(\lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi))$ has strong normal form $(\lambda\phi.\mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi)$ assuming that $\mathbf{S}$ has weak normal form $\mathbf{S}'$.

Example 6.17.

$$\dfrac{\dfrac{}{\lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \to_w \lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)} \qquad \dfrac{\overset{\varphi}{\cdots}}{(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \to_s \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}}{\lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \to_s \lambda\phi.\mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}$$

where φ is

$$\dfrac{\dfrac{}{\lambda\phi'.\phi'\mathsf{c} \to_w \lambda\phi.\phi\mathsf{c}} \qquad \dfrac{}{\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi \to_w \lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi} \qquad \dfrac{\overset{\psi}{\cdots}}{(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\mathsf{c} \to_w \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}}{\dfrac{(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \to_w \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}{(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \to_s \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}}$$

and ψ is as follows

$$\dfrac{\dfrac{}{\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi \to_w \lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi} \qquad \dfrac{}{\mathsf{c} \to_w \mathsf{c}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{gacc} \to_w \mathsf{gacc}} \qquad \dfrac{\overset{\psi''}{\cdots}}{\mathbf{S} \to_w \mathbf{S}'}}{\mathsf{gacc} \to_s \mathsf{gacc} \qquad \mathbf{S} \to_s \mathbf{S}'} \qquad \dfrac{}{\mathsf{c} \to_w \mathsf{c}}}{\mathsf{gacc}\ \mathbf{S} \to_w \mathsf{gacc}\ \mathbf{S}' \qquad \mathsf{c} \to_s \mathsf{c}} \qquad \dfrac{}{\phi \to_w \phi}}{\mathsf{gacc}\ \mathbf{S}\ \mathsf{c} \to_w \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c} \qquad \phi \to_s \phi}}{\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}\ \phi \to_w \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}}{(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\mathsf{c} \to_w \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi}$$

□

Due to the use of exception raising and handling mechanisms, the system lacks the property of strong normalization [Lillibridge, 1999]. However, for all terms obtainable from natural language sentences using the lexical interpretations provided by framework GLχ, a non-terminating evaluation is impossible.

6.3 Cross-Sentential Anaphora

The discourse update function (6.8) is defined in a way that the interpretation of each new sentence is a subterm of the continuation of the current discourse and, moreover, the context in the interpretation of the current discourse is passed (after β-reducing) as an argument to the sentence. This means that all the individuals stored in the context of the interpretation of discourse are available for the interpretation of the sentence. This is a crucial aspect allowing to express cross-sentential anaphora.

The following two examples illustrate in detail how the computation of Discourse (50d) proceeds. They continue Example 6.7 by updating the discourse $(\mathsf{dupd_g}\ \mathbf{D}_0\ \mathbf{S}_1)$, abbreviated below as $\mathbf{D}_1$, computed there. Example 6.18 deals with anaphora related to proper names. Example 6.19 focuses on anaphoric pronouns.

EXAMPLE 6.18. $[\mathsf{dupd_g}\ \mathbf{D}_1\ \mathbf{S}_2]$ Assume Sentence (50b) is uttered after Sentence (50a). The discourse interpretation $\mathbf{D}_1$ (given in (6.16)) is updated with the interpretation $\mathbf{S}_2$ (given in (5.36)) of the new sentence in the following way, where c_{asj} abbreviates $(\mathsf{upd}(\mathbf{about}sj, \mathsf{upd}((\text{named "John"}\,j), \mathsf{upd}(\mathbf{story}s, \mathsf{c}))))$:

$$\begin{aligned}
&\mathsf{dupd_g}\ \mathbf{D}_1\ \mathbf{S}_2\\
&= \lambda\phi.\ \mathbf{D}_1(\lambda e.\mathsf{gacc}\ \mathbf{S}_2\ e\ \phi)\\
&= \lambda\phi.(\lambda\phi'.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}\,j)\wedge\\
&\qquad\qquad (\mathbf{about}sj \wedge \phi'\mathsf{c}_{asj}))))(\lambda e.\mathsf{gacc}\ \mathbf{S}_2\ e\ \phi)\\
&\to_{eval} \lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}\,j)\wedge\\
&\qquad\qquad (\mathbf{about}sj \wedge (\lambda e.\mathsf{gacc}\ \mathbf{S}_2\ e\ \phi)\mathsf{c}_{asj})))\\
&\to_{eval} \lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}\,j)\wedge\\
&\qquad\qquad (\mathbf{about}sj \wedge \mathsf{gacc}\ \mathbf{S}_2\ \mathsf{c}_{asj}\ \phi)))
\end{aligned} \tag{6.17}$$

The computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}_2\ \mathsf{c}_{asj}\ \phi)$ of the term (6.17). According to the definition of gacc, $(\mathbf{S}_2\ \mathsf{c}_{sj}\ \phi)$ is computed firstly:

$$\begin{aligned}
&\mathbf{S}_2\ \mathsf{c}_{sj}\ \phi && \text{(by (6.8b))}\\
&= (\lambda e\phi'.\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e)\wedge\\
&\qquad \phi'(\mathsf{upd}(\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e), e))\mathsf{c}_{asj}\phi\\
&\to^*_{eval} \mathbf{love}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{asj})(\mathsf{sel}(\text{named "Mary"})\mathsf{c}_{asj})\wedge && (6.18)\\
&\qquad \phi(\mathsf{upd}(\mathbf{love}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{asj})(\mathsf{sel}(\text{named "Mary"})\mathsf{c}_{asj}), \mathsf{c}_{asj})
\end{aligned}$$

It is important to note that the environment c_{asj} of $\mathbf{D}_1$ is passed as the first argument (i.e. the context) to $\mathbf{S}_2$. This means that c_{asj} (coming from the interpretation of the discourse being updated) is the current context from which the selection functions of $\mathbf{S}_2$ select individuals. Since c_{asj} already contains the content related to *John*, there is no exception in selecting an individual with the property (named "John") from c_{asj}.

However, there is an exception raised during the attempt to select an individual having the property (named "Mary") in context c_{asj}, because the context does not contain a corresponding variable. The exception is handled by gacc according to Equation (6.8c):

$$\exists m.((\text{named "Mary"}m) \wedge \mathsf{gacc}\ \mathbf{S}_2\ (\mathsf{upd}(\text{named "Mary"}m, \mathsf{c}_{asj}))\phi) \tag{6.19}$$

Abbreviating $(\mathsf{upd}(\text{named "Mary"}m, \mathsf{c}_{asj}))$ by c_{asjm}, the computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}_2\ \mathsf{c}_{asjm}\ \phi)$ of the term shown in Equation (6.19). According to the definition of gacc, $(\mathbf{S}_2\ \mathsf{c}_{asjm}\ \phi)$ is computed firstly:

$$\begin{aligned}
&\mathbf{S}_2\ \mathsf{c}_{asjm}\ \phi && \text{(by (6.8b))}\\
&= (\lambda e\phi'.\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e)\wedge\\
&\qquad \phi'(\mathsf{upd}(\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e), e))\mathsf{c}_{asjm}\phi\\
&\rightarrow^{*}_{eval} \mathbf{love}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{asjm})(\mathsf{sel}(\text{named "Mary"})\mathsf{c}_{asjm})\wedge && (6.20)\\
&\qquad \phi(\mathsf{upd}(\mathbf{love}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{asjm})(\mathsf{sel}(\text{named "Mary"})\mathsf{c}_{asjm}), \mathsf{c}_{asjm})\\
&= \mathbf{love}jm \wedge \phi(\mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{asjm})) && (6.21)
\end{aligned}$$

There is no exception during the computation of $(\mathbf{S}_2\ \mathsf{c}_{asjm}\ \phi)$. Therefore, $(\mathsf{gacc}\ \mathbf{S}_2\ \mathsf{c}_{asjm}\ \phi)$ is evaluated as (6.21).

Finally, combining the β-reductions shown in Equations (6.21), (6.19), (6.17), the result of computation of $(\mathsf{dupd}_{\mathsf{g}}\ \mathbf{D}_1\ \mathbf{S}_2)$ is as follows:

$$\begin{aligned}
&\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}j)\wedge\\
&\qquad (\mathbf{about}sj \wedge \exists(\lambda m.(\text{named "Mary"}m) \wedge \mathbf{love}jm \wedge \phi\mathsf{c}_{asjm})))) && (6.22)
\end{aligned}$$

After unfolding the context c_{asjm}, (6.22) is as follows:

$$\begin{aligned}
&\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}j)\wedge\\
&\qquad \mathbf{about}sj \wedge \exists(\lambda m.(\text{named "Mary"}m) \wedge \mathbf{love}jm\wedge\\
&\qquad\quad \phi(\mathsf{upd}(\mathbf{love}jm,\\
&\qquad\qquad \mathsf{upd}(\text{named "Mary"}m,\\
&\qquad\qquad\ \mathsf{upd}(\mathbf{about}sj,\\
&\qquad\qquad\quad \mathsf{upd}(\text{named "John"}j,\\
&\qquad\qquad\qquad \mathsf{upd}(\mathbf{story}s, \mathsf{c}))))))))))
\end{aligned}$$

□

EXAMPLE 6.19. [$\mathsf{dupd_g}\ \mathbf{D}_2\ \mathbf{S}_3$] The final sentence in Discourse (50d) is (50c), the interpretation $\mathbf{S}_3$ of which is given in Equation (5.40). Interpretation $\mathbf{D}_2$ of the discourse preceding this sentence is computed in Examples 6.7 and 6.18 and is shown in (6.22). To complete the computation of the meaning of (50d), $\mathbf{D}_2$ has to be updated with $\mathbf{S}_3$:

$$\begin{aligned}
&\mathsf{dupd_g}\ \mathbf{D}_2\ \mathbf{S}_3\\
&= \lambda\phi.\ \mathbf{D}_2(\lambda e.\mathsf{gacc}\ \mathbf{S}_3\ e\ \phi)\\
&= \lambda\phi.(\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}j)\wedge\\
&\qquad (\mathbf{about}sj \wedge \exists(\lambda m.(\text{named "Mary"}m) \wedge \mathbf{love}jm \wedge \phi\mathsf{c}_{asjm})))))\\
&\qquad (\lambda e.\mathsf{gacc}\ \mathbf{S}_3\ e\ \phi)\\
\rightarrow_{eval}\ &\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}j)\wedge\\
&\qquad (\mathbf{about}sj \wedge \exists(\lambda m.(\text{named "Mary"}m) \wedge \mathbf{love}jm\wedge\\
&\qquad\quad (\lambda e.\mathsf{gacc}\ \mathbf{S}_3\ e\ \phi)\mathsf{c}_{asjm}))))\\
\rightarrow_{eval}\ &\lambda\phi.\exists(\lambda s.\mathbf{story}s \wedge \exists(\lambda j.(\text{named "John"}j)\wedge \qquad (6.23)\\
&\qquad (\mathbf{about}sj \wedge \exists(\lambda m.(\text{named "Mary"}m) \wedge \mathbf{love}jm \wedge \mathsf{gacc}\ \mathbf{S}_3\ \mathsf{c}_{asjm}\ \phi))))
\end{aligned}$$

The computation continues in the subterm ($\mathsf{gacc}\ \mathbf{S}_3\ \mathsf{c}_{asjm}\ \phi$) of the term (6.23). Recall that c_{asjm} contains formulas (6.5), therefore it can be fully unfolded as follows:

$$\begin{aligned}
\mathsf{c}_{asjm} = \mathsf{upd}(&\mathbf{love}jm,\\
&\mathsf{upd}(\text{named "Mary"}m,\\
&\ \mathsf{upd}(\mathbf{about}sj,\\
&\ \ \mathsf{upd}(\text{named "John"}j,\\
&\ \ \ \mathsf{upd}(\mathbf{story}s, \qquad (6.24)\\
&\ \ \ \ \mathsf{upd}(\forall x.(\text{named "John"}x) \vee (\text{named "Mary"}x) \rightarrow \mathbf{human}x,\\
&\ \ \ \ \ \mathsf{upd}(\forall x.(\text{named "John"}x) \rightarrow \mathbf{male}x,\\
&\ \ \ \ \ \ \mathsf{upd}(\forall x.(\text{named "Mary"}x) \rightarrow \mathbf{female}x))))))))
\end{aligned}$$

Consequently, the following hold (assuming upd to be a conjunction of its arguments):

$$\begin{aligned}
\mathsf{c}_{asjm} &\vdash (\lambda x.\mathbf{male}x \wedge \mathbf{human}x)j\\
\mathsf{c}_{asjm} &\vdash (\lambda x.\mathbf{female}x \wedge \mathbf{human}x)m
\end{aligned}$$

Hence, the selection functions in (6.25) is able to retrieve the individuals j and

m to which the pronouns are anaphorically linked, leading to (6.26):

$$\begin{aligned}
&\mathbf{S}_3\ \mathsf{c}_{asjm}\ \phi && \text{(by (6.8b))}\\
&= (\lambda e\phi'.\mathbf{smile}(\mathsf{sel}(\lambda x.\mathbf{male}x)e)(\mathsf{sel}(\lambda x.\mathbf{female}x)e)\wedge\\
&\qquad \phi'(\mathsf{upd}(\mathbf{smile}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)\\
&\qquad\qquad (\mathsf{sel}(\lambda x.\mathbf{female}x\wedge\mathbf{human}x)e),e))\mathsf{c}_{asjm}\phi\\
&\rightarrow^{*}_{eval} \mathbf{smile}(\mathsf{sel}(\lambda x.\mathbf{male}x)e_{sjm})(\mathsf{sel}(\lambda x.\mathbf{female}x)e_{sjm})\wedge && (6.25)\\
&\qquad \phi(\mathsf{upd}(\mathbf{smile}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)\mathsf{c}_{asjm})\\
&\qquad\qquad (\mathsf{sel}(\lambda x.\mathbf{female}x\wedge\mathbf{human}x)\mathsf{c}_{asjm}),\mathsf{c}_{asjm})\\
&= \mathbf{smile}jm\wedge\phi(\mathsf{upd}(\mathbf{smile}jm,\mathsf{c}_{asjm}) && (6.26)
\end{aligned}$$

Finally, the combination of the β-reductions shown in Equations (6.26) and (6.23) results in the interpretation of Discourse (50d):

$$\begin{aligned}
&\lambda\phi.\exists(\lambda s.\mathbf{story}s\wedge\exists(\lambda j.(\text{named "John"}j)\wedge\\
&\quad(\mathbf{about}sj\wedge\exists(\lambda m.(\text{named "Mary"}m)\wedge\mathbf{love}jm\wedge\mathbf{smile}jm\wedge\\
&\quad\phi(\mathsf{upd}(\mathbf{smile}jm,\mathsf{c}_{asjm}))))) && (6.27)
\end{aligned}$$

Unfolding c_{asjm} (see (6.24)) in (6.27) leads to the following term:

$$\begin{aligned}
&\lambda\phi.\exists(\lambda s.\mathbf{story}s\wedge\exists(\lambda j.(\text{named "John"}j)\wedge\\
&\quad(\mathbf{about}sj\wedge\exists(\lambda m.(\text{named "Mary"}m)\wedge\mathbf{love}jm\wedge\mathbf{smile}jm\wedge\\
&\quad\phi(\mathsf{upd}(\mathbf{love}jm,\\
&\quad\mathsf{upd}(\text{named "Mary"}m,\\
&\quad\mathsf{upd}(\mathbf{about}sj, && (6.28)\\
&\quad\mathsf{upd}(\text{named "John"}j,\\
&\quad\mathsf{upd}(\mathbf{story}s,\\
&\quad\mathsf{upd}(\forall x.(\text{named "John"}x)\vee(\text{named "Mary"}x)\rightarrow\mathbf{human}x,\\
&\quad\mathsf{upd}(\forall x.(\text{named "John"}x)\rightarrow\mathbf{male}x,\\
&\quad\mathsf{upd}(\forall x.(\text{named "Mary"}x)\rightarrow\mathbf{female}x)))))))
\end{aligned}$$

□

Note that the interpretation (6.28) of Discourse (45) is computed in a strictly compositional manner.

6.4 Presupposition Projection in Conditionals

This section applies the framework based on the higher order continuation-based dynamic logic with exceptions to the phenomenon of presupposition projection in

conditionals. Consider Sentences (53a), (54a) and (55). Each of them triggers two kinds of presuppositions: presuppositions of existence of John and a presupposition of existence of John's child. They are similar with respect to the presupposition of existence of John: unless its antecedent is provided by the context, it is accommodated in all three cases. However, they exhibit different presupposition projection behaviour with respect to the presupposition that John has a child: if the context does not provide a referent, the presupposition is accommodated in Sentences (53a) and (54a), but is not in Sentence (55).

(53) a. *If John is back, then his child is happy.*

b. *John has a child.*

(54) a. *If John praises his child, then his child is happy.*

b. *John has a child.*

(55) *If John has a child, then his child is happy.*

$if \ldots then \ldots$ can be interpreted as a dynamic implication (6.29), which, according to Remark 4.29, is an abbreviation for (6.30):

$$[\![if \ldots \widetilde{then} \ldots]\!] = \Rightarrow \tag{6.29}$$

$$\doteq \lambda \mathbf{PQ}.\neg(\mathbf{P}\overline{\wedge}\neg\mathbf{Q}) \tag{6.30}$$

Dynamic conjunction $\overline{\wedge}$ is essential in implementing the desired presupposition projection behaviour, as explained below.

A general observation about presuppositions is that presuppositions are always triggered. When the context in which the presuppositional content is evaluated provides an appropriate antecedent for a presupposition, this presupposition does not emerge and is not accommodated (simply because there is no need for accommodation, as the context already contains equivalent knowledge). This explains the common impression of presupposition "cancelling". In contrast, when the context in which the presuppositional content is evaluated does not provide an appropriate antecedent, the presupposition has to be accommodated.

Assuming that the context does not provide antecedents for presuppositional anaphora, the only possible situation when a presupposition of a conditional does not need to be accommodated is when the antecedent of the conditional asserts a proposition from which the presupposition of the consequent of the conditional can be proved, as in Sentence (55). In all other cases, regardless in which part of the conditional presuppositions are triggered, they have to be accommodated unless contexts in which the sentences are evaluated provide the antecedents.

Therefore, to implement this presupposition projection behaviour in conditionals, the consequent should be evaluated in the context consisting of the context of evaluation of the whole conditional plus the assertion of the antecedent

of the conditional. Dynamic conjunction $\overline{\wedge}$, by its definition, takes care exactly of that. Note that although $\overline{\wedge}$ provides to its continuation the context updated with the conjunction of the logical forms of its two arguments, this contribution is, as desired for conditionals, cancelled by the outermost dynamic negation in $\overline{\neg}(\mathbf{P}\overline{\wedge}\overline{\neg}\mathbf{Q})$.

Consequently, interpretation (6.30) of *if . . . then . . .* implements the following:

- "if"-clause of the conditional is evaluated in the context of evaluation of the whole conditional
- "then"-clause of the conditional is evaluated in the context consisting of the context of evaluation of the whole conditional plus the assertion of the antecedent of the conditional

Table 6.1 gives interpretations for lexical items in sentences (53a), (54a) and (55). Example 6.20 shows meanings of clauses that make up the conditionals. These meanings are computed compositionally from lexical interpretations in Table 6.1.

Consider a discourse interpreted as follows:

$$\mathbf{D} = \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \phi(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))))$$

where context c is the context defined in Example 6.3.

Example 6.21 computes the meaning $\mathbf{S}_1$ of the conditional (53a). Example 6.22 updates $\mathbf{D}$ with $\mathbf{S}_1$. Example 6.23 computes the meaning $\mathbf{S}_2$ of the conditional (54a). Example 6.24 updates $\mathbf{D}$ with $\mathbf{S}_2$. Example 6.25 computes the meaning $\mathbf{S}_3$ of the conditional (55). Example 6.26 updates $\mathbf{D}$ with $\mathbf{S}_3$.

Lexical item	Syntactic category	Dynamic type	Dynamic interpretation in GL
child	n	$\overline{\iota} \to \overline{o}$	$\overline{\mathbf{child}}$
praises	np → np → s	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{praise}}\mathbf{xy}))$
has	np → np → s	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{has}}\mathbf{xy}))$
is_happy	np → s	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\overline{\mathbf{happy}}\mathbf{x})$
is_back	np → s	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\overline{\mathbf{back}}\mathbf{x})$
a	n → np	$(\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o})$	$\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx})$
John	np	$(\overline{\iota} \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named “John”}))$
he	np	$(\overline{\iota} \to \overline{o}) \to \overline{o}$	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x))$
's	np → n → np	$((\overline{\iota} \to \overline{o}) \to \overline{o}) \to (\overline{\iota} \to \overline{o}) \to ((\overline{\iota} \to \overline{o}) \to \overline{o})$	$\lambda\mathbf{YX}.\lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\mathbf{Xx}) \overline{\wedge} \mathbf{Y}(\overline{[\![has]\!]}\mathbf{x}))))$

Table 6.1: Dynamic lexical interpretations in framework GL.

Example 6.20. Normalized meanings of *his child is happy*, *John is back*, *John praises his child* and *John has a child* are respectively as follows:

$$\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}) \to^*_{eval} \lambda e\phi.\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi(\mathsf{upd}(\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e), e))$$

$$\overline{[\![is_back]\!]}\widetilde{[\![John]\!]} \to^*_{eval} \lambda e\phi.\mathbf{back}(\mathsf{sel}(\text{named “John”})e) \wedge \phi(\mathsf{upd}(\mathbf{back}(\mathsf{sel}(\text{named “John”})e), e))$$

$$(\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]} \to^*_{eval} \lambda e\phi.\mathbf{praise}(\mathsf{sel}(\text{named “John”})e)(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)y)e) \wedge \phi(\mathsf{upd}(\mathbf{praise}(\mathsf{sel}(\text{named “John”})e)(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)y)e), e))$$

$$\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]} \to^*_{eval} \lambda e\phi.\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}\ (\mathsf{sel}\ (\text{named “John”})\ (\mathsf{upd}(\mathbf{child}y, e)))\ y \wedge \phi(\mathsf{upd}(\mathbf{has}\ (\mathsf{sel}\ (\text{named “John”})\ (\mathsf{upd}(\mathbf{child}y, e)))\ y, \mathsf{upd}(\mathbf{child}y, e))))$$

□

EXAMPLE 6.21. [*If John is back, then his child is happy*, $\mathbf{S}_1$] The meaning $\mathbf{S}_1$ of (53a) is computed by β-reducing term (6.31):

$$\begin{aligned}\mathbf{S}_1 &= [\![if \ldots \widetilde{then} \ldots]\!](\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \\ &= (\lambda\mathbf{PQ}.\overline{\neg}(\mathbf{P}\overline{\wedge}\overline{\neg}\mathbf{Q}))(\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \\ \rightarrow^*_{eval} &\ \overline{\neg}((\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))\end{aligned} \tag{6.31}$$

Below the term $\overline{\neg}((\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))$ is normalized, resulting in the final interpretation (6.32) of the conditional.

$$\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \rightarrow^*_{eval} \lambda e\phi.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e$$

$$\begin{aligned}&(\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \\ =\ &(\lambda\mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi))(\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})(\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))) \\ \rightarrow^*_{eval}\ &\lambda e\phi.(\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})e(\lambda e.(\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))e\phi) \\ =\ &\lambda e\phi.(\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})e(\lambda e.(\lambda e\phi.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e)e\phi) \\ \rightarrow^*_{eval}\ &\lambda e\phi.(\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})e(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e) \\ =\ &\lambda e\phi.(\lambda e\phi.\mathbf{back}(\mathsf{sel}(\text{named “John”})e)\wedge \\ &\phi(\mathsf{upd}(\mathbf{back}(\mathsf{sel}(\text{named “John”})e),e)))e(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e) \\ \rightarrow^*_{eval}\ &\lambda e\phi.\mathbf{back}(\mathsf{sel}(\text{named “John”})e)\wedge \\ &(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e)(\mathsf{upd}(\mathbf{back}(\mathsf{sel}(\text{named “John”})e),e)) \\ \rightarrow_{eval}\ &\lambda e\phi.\mathbf{back}(\mathsf{sel}(\text{named “John”})e) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_b)x)e_b) \wedge \phi e_b\end{aligned}$$

where $e_b = \mathsf{upd}(\mathbf{back}(\mathsf{sel}(\text{named "John"})e), e)$

$$
\begin{aligned}
&\overline{\neg}((\overline{[\![is_back]\!]}\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))\\
&= \overline{\neg}(\lambda e\phi.\mathbf{back}(\mathsf{sel}(\text{named "John"})e) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_b)x)e_b) \wedge \phi e_b)\\
&= \lambda e\phi.\neg(\mathbf{back}(\mathsf{sel}(\text{named "John"})e) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_b)x)e_b)) \wedge \phi e \qquad (6.32)
\end{aligned}
$$

□

EXAMPLE 6.22. [$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_1$] Assume that Sentence (53a), interpretation $\mathbf{S}_1$ of which is computed in the previous example, is uttered in discourse interpreted in the following way:

$$\mathbf{D} = \lambda\phi.\exists(\lambda j.(\text{named "John"}j \wedge \phi(\mathsf{upd}(\text{named "John"}j, \mathsf{c}))))$$

Context c is the context defined in Example 6.3. The update of $\mathbf{D}$ with $\mathbf{S}_1$ is then computed as follows:

$$
\begin{aligned}
\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_1 &= \lambda\phi.\mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}_1\ e\ \phi)\\
&= \lambda\phi.(\lambda\phi.\exists(\lambda j.(\text{named "John"}j \wedge \phi(\mathsf{upd}(\text{named "John"}j, \mathsf{c})))))(\lambda e.\mathsf{gacc}\ \mathbf{S}_1\ e\ \phi)\\
&\rightarrow_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}j \wedge (\lambda e.\mathsf{gacc}\ \mathbf{S}_1\ e\ \phi)(\mathsf{upd}(\text{named "John"}j, \mathsf{c}))))\\
&\rightarrow_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}j \wedge \mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\text{named "John"}j, \mathsf{c}))\ \phi)) \qquad (6.33)
\end{aligned}
$$

The computation continues in the subterm ($\mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\text{named "John"}j, \mathsf{c}))\ \phi$) of (6.33). ($\mathbf{S}_1\ (\mathsf{upd}(\text{named "John"}j, \mathsf{c}))\ \phi$) is computed firstly:

$$
\begin{aligned}
&\mathbf{S}_1\ (\mathsf{upd}(\text{named "John"}j, \mathsf{c}))\ \phi\\
&= (\lambda e\phi.\neg(\mathbf{back}(\mathsf{sel}(\text{named "John"})e) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_b)x)e_b)) \wedge \phi e)\\
&\quad (\mathsf{upd}(\text{named "John"}j, \mathsf{c}))\ \phi\\
&\rightarrow^*_{eval} \neg(\mathbf{back}(\mathsf{sel}(\text{named "John"})(\mathsf{upd}(\text{named "John"}j, \mathsf{c})))\wedge \qquad (6.34)\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{bj})x)\mathsf{c}_{bj})) \wedge \phi(\mathsf{upd}(\text{named "John"}j, \mathsf{c}))
\end{aligned}
$$

where $\mathsf{c}_{bj} = \mathsf{upd}(\mathbf{back}(\mathsf{sel}(\text{named “John”})(\mathsf{upd}(\text{named “John”}j, \mathsf{c}))), \mathsf{upd}(\text{named “John”}j, \mathsf{c}))$. Clearly, the following holds:

$$\text{named “John”}j, \mathsf{c} \vdash \text{named “John”}j$$

Therefore, individual j satisfying property (named “John”) is selected by sel from (upd(named “John”j, c)) and c_{bj} can be unfolded as follows:

$$\begin{aligned}\mathsf{c}_{bj} = \; & \mathbf{back}j \wedge (\text{named “John”}j) \wedge \\ & (\forall x.(\text{named “John”}x) \vee (\text{named “Mary”}x) \rightarrow \mathbf{human}x) \wedge \\ & (\forall x.(\text{named “John”}x) \rightarrow \mathbf{male}x) \wedge \\ & (\forall x.(\text{named “Mary”}x) \rightarrow \mathbf{female}x)\end{aligned}$$

Moreover, since the following proofs can be found

$$\begin{aligned}&\mathsf{c}_{bj} \vdash \text{named “John”}j \\ &\mathsf{c}_{bj} \vdash (\lambda y.\mathbf{male}y \wedge \mathbf{human}y)j\end{aligned}$$

the selection function sel in (6.34) retrieves the individual j from c_{bj} based on the properties (named “John”) and $(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)$, leading to (6.35):

$$\neg(\mathbf{back}j \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{c}_{bj})) \wedge \phi(\mathsf{upd}(\text{named “John”}j, \mathsf{c})) \tag{6.35}$$

However, there is an exception in the term $(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{c}_{bj})$, because c_{bj} does not contain a proposition that John has a child, i.e.

$$\text{there is no } \mathsf{a} \text{ such that } \mathsf{c}_{bj} \vdash (\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{a}$$

Therefore, the presupposition that there exists a child of John is accommodated according to the exception handler (7.12c):

$$\begin{aligned} & \exists(\lambda h.((\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)h) \wedge \mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}((\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)h), \mathsf{upd}(\text{named "John"}j, \mathsf{c}))\ \phi) \\ \rightarrow^*_{eval}\ & \exists(\lambda h.(\mathbf{child}h \wedge \mathbf{has}jh) \wedge \mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi) \end{aligned} \tag{6.36}$$

The computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}_1\ (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi)$ of (6.36):

$$\begin{aligned} & \mathbf{S}_1\ (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi \\ = & (\lambda e\phi.\neg(\mathbf{back}(\mathsf{sel}(\text{named "John"})e) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_b)x)e_b)) \wedge \phi e) \\ & (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi \\ \rightarrow^*_{eval}\ & \neg(\mathbf{back}(\mathsf{sel}(\text{named "John"})(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c}))))\wedge \\ & \quad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{bjc})x)\mathsf{c}_{bjc}))\wedge \\ & \quad \phi(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c}))) \end{aligned} \tag{6.37}$$

where

$$\begin{aligned} \mathsf{c}_{bjc} = \mathsf{upd}(\mathbf{back}(\mathsf{sel}(\text{named "John"})(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))), \\ \mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c}))) \end{aligned}$$

Clearly, the following holds:

$$\mathbf{child}h \wedge \mathbf{has}jh, \text{named "John"}j, \mathsf{c} \vdash \text{named "John"}j$$

Therefore, c_{bjc} can be unfolded and presented in a more clear form as follows:

$$\begin{aligned} \mathsf{c}_{bjc} = & \mathbf{back}j \wedge \mathbf{child}h \wedge \mathbf{has}jh \wedge (\text{named "John"}j)\wedge \\ & (\forall x.(\text{named "John"}x) \vee (\text{named "Mary"}x) \rightarrow \mathbf{human}x)\wedge \\ & (\forall x.(\text{named "John"}x) \rightarrow \mathbf{male}x)\wedge \\ & (\forall x.(\text{named "Mary"}x) \rightarrow \mathbf{female}x) \end{aligned}$$

Moreover, as the following derivations can be obtained

$$\mathsf{c}_{bjc} \vdash (\lambda y.\mathbf{male}y \wedge \mathbf{human}y)j$$
$$\mathsf{c}_{bjc} \vdash (\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)h$$

(6.37) leads to the following term:

$$\neg(\mathbf{back}j \wedge \neg\mathbf{happy}h) \wedge \phi(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c}))) \tag{6.38}$$

After composing terms (6.38), (6.36), (6.33) and (6.35) the final interpretation of the initial discourse updated with sentence (53a) (*If John is back, then his child is happy*) is obtained:

$$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_1 \rightarrow^*_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}j \wedge \exists(\lambda h.(\mathbf{child}h \wedge \mathbf{has}jh) \wedge \neg(\mathbf{back}j \wedge \neg\mathbf{happy}h) \wedge \phi(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))))) \tag{6.39}$$

Note that the resulting interpretation (6.39) contains the content of $\mathbf{D}$ (i.e. an existentially quantified variable j for and an individual named "John" in the logical formula and in the context, and common knowledge c in the context) and the accommodated content related to presupposition of existence of John's child (i.e. an existentially quantified variable h satisfying the property of being John's child) in the logical formula and in the context. □

Example 6.23. [*If John praises his child, then his child is happy*, $\mathbf{S}_2$] The meaning $\mathbf{S}_2$ of (54a) is computed by β-reducing term (6.40):

$$\begin{aligned}\mathbf{S}_2 &= [\![\widetilde{if \ldots then} \ldots]\!](\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})) \\ &= \overline{\neg}(\mathbf{P}\overline{\wedge}\overline{\neg}\mathbf{Q})((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})) \\ &= \overline{\neg}(((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})))\end{aligned} \tag{6.40}$$

Below the term $\overline{\neg}(((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})))$ is normalized, leading to the final interpretation (6.41) of the conditional.

$$\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})) \rightarrow^*_{eval} \lambda e\phi.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e$$

$$
\begin{aligned}
&((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})\overline{\wedge}\neg(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]}))\\
&= (\lambda\mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi))((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})(\neg(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})))\\
&\rightarrow^*_{eval} \lambda e\phi.((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})e(\lambda e.(\neg(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})))e\phi)\\
&= \lambda e\phi.((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})\\
&\qquad e(\lambda e.(\lambda e\phi.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e)x)e)\wedge\phi e)e\phi)\\
&\rightarrow^*_{eval} \lambda e\phi.((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})e(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e)x)e)\wedge\phi e)\\
&= \lambda e\phi.(\lambda e\phi.\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e)\wedge\\
&\qquad\qquad \phi(\mathsf{upd}(\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e),e)))\\
&\qquad e(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e)x)e)\wedge\phi e)\\
&\rightarrow^*_{eval} \lambda e\phi.\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e)\wedge\\
&\qquad (\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e)x)e)\wedge\phi e)\\
&\qquad (\mathsf{upd}(\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e),e))\\
&\rightarrow^*_{eval} \lambda e\phi.\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e)\wedge\\
&\qquad\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e_p)x)e_p)\wedge\phi e_p
\end{aligned}
$$

where $e_p = \mathsf{upd}(\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e),e)$

$$
\begin{aligned}
&\neg(((\overline{[\![praises]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))\widetilde{[\![John]\!]})\overline{\wedge}\neg(\overline{[\![is_happy]\!]}(\widetilde{[\![his]\!]}\overline{[\![child]\!]})))\\
&\rightarrow^*_{eval} \neg(\lambda e\phi.\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e)\wedge\\
&\qquad\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e_p)x)e_p)\wedge\phi e_p)\\
&\rightarrow^*_{eval} \lambda e\phi.\neg(\mathbf{praise}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\lambda y.\mathbf{child}y\wedge\mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x\wedge\mathbf{human}x)e)y)e)\wedge\\
&\qquad\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x\wedge\mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y\wedge\mathbf{human}y)e_p)x)e_p))\wedge\phi e
\end{aligned}
\tag{6.41}
$$

□

EXAMPLE 6.24. [$\mathsf{dupd_g}$ **D** $\mathbf{S}_2$] Assume that Sentence (54a), the interpretation $\mathbf{S}_2$ of which is computed in the previous example, is uttered in discourse interpreted in the following way:

$$\mathbf{D} = \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \phi(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))) \tag{6.42}$$

Context c is the context defined in Example 6.3. The update of **D** with $\mathbf{S}_2$ is computed in the following way:

$$\begin{aligned}
\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_2 &= \lambda\phi.\mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}_2\ e\ \phi) \\
&= \lambda\phi.(\lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \phi(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))))(\lambda e.\mathsf{gacc}\ \mathbf{S}_2\ e\ \phi) \\
&\rightarrow_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge (\lambda e.\mathsf{gacc}\ \mathbf{S}_2\ e\ \phi)(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))) \\
&\rightarrow_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \mathsf{gacc}\ \mathbf{S}_2\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\ \phi))
\end{aligned} \tag{6.43}$$

The computation continues in the subterm (gacc **S** (upd(named "John"j, c)) ϕ) of (6.43):

$$\begin{aligned}
&\mathbf{S}_2\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\ \phi \\
&= (\lambda e\phi.\neg(\mathbf{praise}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)y)e) \wedge \\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_p)x)e_p)) \wedge \phi e) \\
&\qquad\qquad (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\phi \\
&\rightarrow^*_{eval} \neg(\mathbf{praise}(\mathsf{sel}(\text{named "John"})\mathsf{c}_j)(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)\mathsf{c}_j)y)\mathsf{c}_j) \wedge \\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{pj})x)\mathsf{c}_{pj})) \wedge \phi\mathsf{c}_j
\end{aligned} \tag{6.44}$$

where

$$\begin{aligned}
\mathsf{c}_j &= \mathsf{upd}(\text{named "John"}\, j, \mathsf{c}) \\
\mathsf{c}_{pj} &= \mathsf{upd}(\mathbf{praise}(\mathsf{sel}\ (\text{named "John"})\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))) \\
&\qquad (\mathsf{sel}\ (\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}\ (\lambda x.\mathbf{male}x \wedge \mathbf{human}x)\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))y)\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))), \\
&\qquad\qquad \mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))
\end{aligned}$$

Clearly, the following holds:

$$\text{named “John”}j, \mathsf{c} \vdash \text{ named “John”}j$$

Therefore, individual j satisfying property (named “John”) is selected by sel from (upd(named “John”j, c)) and all subterms (sel(named “John”)c_j) can be substituted with j. Then, c_{pj} is as follows (assuming upd is a conjunction):

$$\begin{aligned}\mathsf{c}_{pj} = \mathbf{praise}j(\mathsf{sel}\ (\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)\ (\mathsf{upd}(\text{named “John”}j, \mathsf{c})) \wedge (\text{named “John”}j) \wedge \\ (\forall x.(\text{named “John”}x) \vee (\text{named “Mary”}x) \rightarrow \mathbf{human}x) \wedge \\ (\forall x.(\text{named “John”}x) \rightarrow \mathbf{male}x) \wedge \\ (\forall x.(\text{named “Mary”}x) \rightarrow \mathbf{female}x)\end{aligned}$$

Moreover, since the following holds

$$\mathsf{c}_{pj} \vdash (\lambda y.\mathbf{male}y \wedge \mathbf{human}y)j$$

all subterms ($\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{pj}$) in (6.44) can be substituted with j leading to (6.45):

$$\neg(\mathbf{praise}j(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)\mathsf{c}_j) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{c}_{pj})) \wedge \phi\mathsf{c}_j \tag{6.45}$$

However, terms ($\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}yj)\mathsf{c}_j$) and ($\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{c}_{pj}$) raise exceptions, because

$$\begin{aligned}&\text{there is no } \mathsf{a} \text{ such that } \mathsf{c}_j \vdash\ (\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)\mathsf{a} \\ &\text{there is no } \mathsf{a} \text{ such that } \mathsf{c}_{pj} \vdash\ (\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{a}\end{aligned}$$

Therefore, the presupposition that there exists a child of John is accommodated according to the exception handler (7.12c):

$$\begin{aligned}&\exists(\lambda h.((\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)h) \wedge \mathsf{gacc}\ \mathbf{S}_2\ (\mathsf{upd}((\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)h), \mathsf{upd}(\text{named “John”}j, \mathsf{c})))\ \phi) \\ \rightarrow^*_{eval}\ &\exists(\lambda h.(\mathbf{child}h \wedge \mathbf{has}jh) \wedge \mathsf{gacc}\ \mathbf{S}_2\ (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named “John”}j, \mathsf{c})))\ \phi)\end{aligned} \tag{6.46}$$

The computation continues in the subterm ($\mathsf{gacc}\ \mathbf{S}_2\ (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi$) of (6.46):

$$\begin{aligned}
&\mathbf{S}_2\ (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi \\
=\ &(\lambda e\phi.\neg(\mathbf{praise}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)e)y)e) \wedge \\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_p)x)e_p)) \wedge \phi e) \\
&\qquad (\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))\ \phi \\
\rightarrow^*_{eval}\ &\neg(\mathbf{praise}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{jc})(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)\mathsf{c}_{jc})y)\mathsf{c}_{jc}) \wedge \\
&\quad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{pjc})x)\mathsf{c}_{pjc})) \wedge \phi\mathsf{c}_{jc} \\
=\ &\neg(\mathbf{praise}j(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)\mathsf{c}_{jc}) \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{c}_{pjc})) \wedge \phi\mathsf{c}_{jc} \qquad (6.47)
\end{aligned}$$

where

$$\begin{aligned}
\mathsf{c}_{jc} &= \mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})) \\
\mathsf{c}_{pjc} &= \mathsf{upd}(\mathbf{praise}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{jc})(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\lambda x.\mathbf{male}x \wedge \mathbf{human}x)\mathsf{c}_{jc})y)\mathsf{c}_{jc}), \mathsf{c}_{jc}) \\
&= \mathsf{upd}(\mathbf{praise}j(\mathsf{sel}(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)\mathsf{c}_{jc}), \mathsf{c}_{jc})
\end{aligned}$$

Since the following derivations hold

$$\begin{aligned}
\mathsf{c}_{jc} &\vdash\ (\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)h \\
\mathsf{c}_{pjc} &\vdash\ (\lambda y.\mathbf{child}y \wedge \mathbf{has}jy)h
\end{aligned}$$

term (6.47) can be rewritten as follows:

$$\neg(\mathbf{praise}jh \wedge \neg\mathbf{happy}h) \wedge \phi(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}j, \mathsf{c})))) \qquad (6.48)$$

Finally, the composition of terms (6.43), (6.46), (6.48) results in the interpretation of the initial discourse updated with the conditional (54a) (*If John praises his child, then his child is happy*):

$$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_2 \rightarrow^*_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \exists(\lambda h.(\mathbf{child}h \wedge \mathbf{has}jh) \wedge \underbrace{\neg(\mathbf{praise}jh \wedge \neg\mathbf{happy}h)}_{\text{assertion of the sentence}} \wedge$$
$$\phi\underbrace{(\mathsf{upd}(\mathbf{child}h \wedge \mathbf{has}jh, \mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))}_{\text{context}})))))) \tag{6.49}$$

Note that, analogously to the case in Example (6.22), interpretation (6.49) contains the content of **D** and the accommodated content related to presupposition of existence of John's child in the logical formula and in the context. □

EXAMPLE 6.25. [*If John has a child, then his child is happy*, $\mathbf{S}_3$] The meaning $\mathbf{S}_3$ of (55) is computed by β-reducing term (6.50):

$$\begin{aligned}
\mathbf{S}_3 &= [\![if \ldots \widetilde{then} \ldots]\!](\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \\
&= \overline{\neg}(\mathbf{P}\overline{\wedge}\overline{\neg}\mathbf{Q})(\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \\
\rightarrow^*_{eval} &\ \overline{\neg}((\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))
\end{aligned} \tag{6.50}$$

Below the term $\overline{\neg}((\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))$ is normalized, leading to the final interpretation (6.51) of the conditional.

$$\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \rightarrow^*_{eval} \lambda e\phi.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e$$

$$\begin{aligned}
&(\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})\overline{\wedge}\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})) \\
= &\ (\lambda\mathbf{AB}.\lambda e\phi.\mathbf{A}e(\lambda e.\mathbf{B}e\phi))(\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})(\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]}))) \\
\rightarrow^*_{eval} &\ \lambda e\phi.(\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})e(\lambda e.(\overline{\neg}(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))e\phi)
\end{aligned}$$

$$
\begin{aligned}
&= \lambda e\phi.(\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})e(\lambda e.(\lambda e\phi.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e)e\phi)\\
\rightarrow^*_{eval}\ &\lambda e\phi.(\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})e(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e)\\
\rightarrow^*_{eval}\ &\lambda e\phi.(\lambda e\phi.\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad \phi(\mathsf{upd}(\mathbf{has}(\mathsf{sel}(\text{named ``John''})\mathsf{upd}(\mathbf{child}y, e))y, \mathsf{upd}(\mathbf{child}y, e)))))\\
&\quad e(\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e)\\
\rightarrow^*_{eval}\ &\lambda e\phi.\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad (\lambda e.\neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e)x)e) \wedge \phi e)\\
&\qquad (\mathsf{upd}(\mathbf{has}(\mathsf{sel}(\text{named ``John''})\mathsf{upd}(\mathbf{child}y, e))y, \mathsf{upd}(\mathbf{child}y, e))))\\
\rightarrow^*_{eval}\ &\lambda e\phi.\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_c)x)e_c) \wedge \phi e_c)
\end{aligned}
$$

where $e_c = \mathsf{upd}(\mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y, \mathsf{upd}(\mathbf{child}y, e))$

$$
\begin{aligned}
&\neg((\overline{[\![has]\!]}(\overline{[\![a]\!]}\ \overline{[\![child]\!]})\widetilde{[\![John]\!]})\overline{\wedge}\neg(\overline{[\![is_happy]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![child]\!]})))\\
\rightarrow^*_{eval}\ &\neg(\lambda e\phi.\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_c)x)e_c) \wedge \phi e_c))\\
\rightarrow^*_{eval}\ &\lambda e\phi.\neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_c)x)e_c) \wedge \top)) \wedge \phi e\\
\equiv\ &\lambda e\phi.\neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named ``John''})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_c)x)e_c))) \wedge \phi e
\end{aligned}
\tag{6.51}
$$

□

EXAMPLE 6.26. [$\mathsf{dupd_g}$ **D** $\mathbf{S}_3$] Assume that Sentence (55), interpreted as $\mathbf{S}_3$ in the previous example, is uttered in discourse interpreted in the following way:

$$
\mathbf{D} = \lambda\phi.\exists(\lambda j.(\text{named ``John''}j \wedge \phi(\mathsf{upd}(\text{named ``John''}j, \mathsf{c}))))
\tag{6.52}
$$

Context c is the context defined in Example 6.3. The update of $\mathbf{D}$ with $\mathbf{S}_3$ is computed in the following way:

$$\begin{aligned}
\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_3 &= \lambda\phi.\mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}_3\ e\ \phi)\\
&= \lambda\phi.(\lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \phi(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))))(\lambda e.\mathsf{gacc}\ \mathbf{S}_3\ e\ \phi)\\
&\rightarrow_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge (\lambda e.\mathsf{gacc}\ \mathbf{S}_3\ e\ \phi)(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))))\\
&\rightarrow_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}\, j \wedge \mathsf{gacc}\ \mathbf{S}_3\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\ \phi)) \qquad (6.53)
\end{aligned}$$

The computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}_3\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\ \phi)$ of (6.53):

$$\begin{aligned}
&\mathbf{S}_3\ (\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\ \phi\\
&= (\lambda e\phi.\neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named "John"})(\mathsf{upd}(\mathbf{child}y, e)))y \wedge\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)e_c)x)e_c))) \wedge \phi e)(\mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))\ \phi\\
&\rightarrow^*_{eval} \neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}(\mathsf{sel}(\text{named "John"})(\mathsf{upd}(\mathbf{child}y, \mathsf{c}_j)))y \wedge \qquad (6.54)\\
&\qquad \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{jc})x)\mathsf{c}_{jc}))) \wedge \phi\mathsf{c}_j
\end{aligned}$$

where

$$\mathsf{c}_j = \mathsf{upd}(\text{named "John"}\, j, \mathsf{c})$$

$$\mathsf{c}_{jc} = \mathsf{upd}(\mathbf{has}(\mathsf{sel}\ (\text{named "John"})\ (\mathsf{upd}(\mathbf{child}y, \mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))\ y, \mathsf{upd}(\mathbf{child}y, \mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))$$

Clearly, the following derivation holds:

$$\mathbf{child}y, \text{named "John"}\, j, \mathsf{c} \vdash\ \text{named "John"}\, j$$

Therefore, individual j satisfying property (named "John") is selected by sel from $(\mathsf{upd}(\mathbf{child}y, \mathsf{upd}(\text{named "John"}\, j, \mathsf{c})))$ and all subterms $(\mathsf{sel}(\text{named "John"})(\mathsf{upd}(\mathbf{child}y, \mathsf{upd}(\text{named "John"}\, j, \mathsf{c}))))$ in (6.54) can be substituted with j. Then, c_{jc} is as follows (assuming upd is conjunction):

$$\begin{aligned}
\mathsf{c}_{jc} = \mathbf{has}jy \wedge \mathbf{child}y \wedge (\text{named "John"}\, j) \wedge\\
(\forall x.(\text{named "John"}\, x) \vee (\text{named "Mary"}\, x) \rightarrow \mathbf{human}x) \wedge\\
(\forall x.(\text{named "John"}\, x) \rightarrow \mathbf{male}x) \wedge\\
(\forall x.(\text{named "Mary"}\, x) \rightarrow \mathbf{female}x)
\end{aligned}$$

Moreover, the following derivation holds:

$$\mathsf{c}_{jc} \vdash (\lambda y.\mathbf{male}y \wedge \mathbf{human}y)j$$

Therefore, all subterms $(\mathsf{sel}(\lambda y.\mathbf{male}y \wedge \mathbf{human}y)\mathsf{c}_{jc})$ in (6.54) can be substituted with j leading to (6.55):

$$\neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy \wedge \neg\mathbf{happy}(\mathsf{sel}(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)\mathsf{c}_{jc}))) \wedge \phi(\mathsf{upd}(\text{named "John"}j, \mathsf{c})) \tag{6.55}$$

Importantly, there is no exception in term (6.55): the individual y satisfying the property $(\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)$ is selected from c_{jc}, as the following derivation holds:

$$\mathsf{c}_{jc} \vdash \ (\lambda x.\mathbf{child}x \wedge \mathbf{has}jx)y$$

This very moment is often regarded as presupposition "cancelling". In fact, there is no cancelling as such. The content that is presupposed is simply already contained in the context over which the presupposition is evaluated, and therefore the required referent can be selected. In conditionals the content of the antecedent of the conditional is contained in the context over which the consequent is evaluated. And since the antecedent of Sentence (55) conveys *John has a child* (i.e. its interpretation creates an existentially quantified variable y such that it satisfies the property of being a child of someone named "John"), the individual variable y (standing for John's child) could be selected in the consequent. Consequently, term (6.55) can be rewritten as follows:

$$\neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy \wedge \neg\mathbf{happy}y)) \wedge \phi(\mathsf{upd}(\text{named "John"}j, \mathsf{c})) \tag{6.56}$$

The composition of terms (6.53) and (6.56) leads to the final interpretation of the initial discourse updated with Sentence (55) (*If John has a child, then his child is happy*):

$$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}_3 \rightarrow^*_{eval} \lambda\phi.\exists(\lambda j.(\text{named "John"}j \wedge \neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy \wedge \neg\mathbf{happy}y)) \wedge \phi(\mathsf{upd}(\text{named "John"}j, \mathsf{c})))) \tag{6.57}$$

Note that, analogously to the case in Examples (6.22) and (6.24), interpretation (6.57) contains the content of **D** and the assertion of the conditional (55) (i.e. $\neg(\exists(\lambda y.\mathbf{child}y \wedge \mathbf{has}jy \wedge \neg\mathbf{happy}y)))$ in the logical formula. However, in contrast to (6.39) and (6.57), (6.57) does not contain any accommodated content since there was no exception raised (and, hence, handled) as the antecedent of the conditional contained the content (*John has a child*) that was accessible to the presupposition triggering expression (*his child*) in the consequent. □

Lexical item	Syntactic category	Dynamic interpretation in GL
boxer	n	$\overline{\mathbf{boxer}}$
apartment	n	$\overline{\mathbf{ap}}$
escaped_from	np → np → s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{esc}}\mathbf{xy}))$
poss	np → np → s	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{poss}}\mathbf{xy}))$
a	n → np	$\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx})$
he	np	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{male}x))$
's	np → n → np	$\lambda\mathbf{YX}.\lambda\mathbf{P}.\mathbf{P}(\widetilde{\mathsf{sel}}(\lambda\mathbf{x}.((\mathbf{Xx})\overline{\wedge}\mathbf{Y}(\overline{[\![poss]\!]}\mathbf{x}))))$

Table 6.2: Dynamic lexical interpretations in framework GL

6.5 Binding Problem

Another challenging issue related to presuppositions is the **binding problem**. When Sentence (56), for example, is uttered on its own, it is generally understood that the boxer escaped from the apartment that belongs to this very boxer. In other words, the presupposition triggered by the noun phrase *his apartment* is bound to the boxer introduced in the same sentence.

(56) *A boxer escaped from his apartment.*

However, it is not a trivial task to formally achieve this binding requirement.[3] Example 6.27 sketches the computation of the meaning **S** of Sentence (56) using dynamic lexical interpretations developed so far and Example (6.28) illustrates the binding problem in detail. This problem can, however, be fixed by adding an exception handler to the interpretation of the dynamic existential quantifier. Example 6.30 computes the meaning $\mathbf{S}'$ of the sentence using such a refined interpretation of the dynamic existential quantifier and Example 6.31 shows that with this new interpretation the binding problem is avoided.

Example 6.27. [**S**] Given lexical interpretations presented in Table 6.2, the interpretation of Sentence (56) can be computed and its normal form is shown in (6.61). Some intermediary steps of the computation, particularly normal forms of *his apartment*, *escaped from his apartment* and *a boxer* are shown in (6.58), (6.59) and (6.60) respectively:

$$\begin{aligned} & \widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![apartment]\!]} \\ \rightarrow^{*}_{eval}\ & \lambda\mathbf{P}.\mathbf{P}(\lambda e.\mathsf{sel}\ (\lambda x.\mathbf{ap}x \wedge \mathbf{poss}\ (\mathsf{sel}\ (\lambda y.\mathbf{male}y)\ e)\ x)\ e) \end{aligned} \tag{6.58}$$

[3] DRT's account, for example, is not satisfactory as it uses preliminary indexing for solving the anaphor within the sentence and, due to this indexing, it artificially restricts (with the accessibility constraint stating that no variable should become unbound) placing the presuppositional content outside the scope of the existential quantifier.

$$\begin{aligned}&\overline{[\![escaped_from]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![apartment]\!]})\\ \rightarrow^*_{eval}\ &\lambda\mathbf{Z}.\mathbf{Z}(\lambda\mathbf{z}.(\lambda e\phi.\underbrace{\mathbf{esc}(\mathbf{z}e)(\mathsf{sel}(\lambda x.\mathbf{ap}x\wedge\mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e)x)e)}_{\psi}\wedge\\ &\qquad\phi(\mathsf{upd}(\psi,e))))\end{aligned}\tag{6.59}$$

$$\overline{[\![a]\!]}\ \overline{[\![boxer]\!]}\rightarrow^*_{eval}\lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda x.\mathbf{boxer}x\wedge\mathbf{Q}(\lambda e'.x)(\mathsf{upd}(\mathbf{boxer}x,e))\phi)\tag{6.60}$$

$$\begin{aligned}&\mathbf{S}=\overline{[\![escaped_from]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![apartment]\!]})(\overline{[\![a]\!]}\ \overline{[\![boxer]\!]})\\ \rightarrow^*_{eval}\ &\lambda e\phi.\exists(\lambda x.\mathbf{boxer}x\wedge\underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z\wedge\mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e_b)z)e_b)}_{\psi}\wedge\\ &\qquad\phi(\mathsf{upd}(\psi,e_b)))\end{aligned}\tag{6.61}$$

where $e_b=\mathsf{upd}(\mathbf{boxer}x,e)$ □

Example 6.28. [$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}$, binding problem] Assume Sentence (56) is uttered in a discourse with empty content interpreted as $(\lambda\phi.\phi\mathsf{c})$, where $\mathsf{c}=\forall x.(\mathbf{boxer}x\rightarrow\mathbf{male}x)$. The interpretation of the resulting discourse is computed as follows:

$$\begin{aligned}\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}&=\lambda\phi.\mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)&&\text{(by (6.8a))}\\ &=\lambda\phi.(\lambda\phi'.\phi'\mathsf{c})(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\\ &\rightarrow_{eval}\lambda\phi.(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\mathsf{c}\\ &\rightarrow_{eval}\lambda\phi.\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}\ \phi&&(6.62)\end{aligned}$$

The computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}\ \phi)$ of (6.62). $(\mathbf{S}\ \mathsf{c}\ \phi)$ is evaluated as follows:

$$\begin{aligned}&\mathbf{S}\ \mathsf{c}\ \phi\\ &\quad=(\lambda e\phi.\exists(\lambda x.\mathbf{boxer}x\wedge\underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z\wedge\mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e_b)z)e_b)}_{\psi}\wedge\\ &\qquad\phi(\mathsf{upd}(\psi,e_b))))\ \mathsf{c}\ \phi\\ &\rightarrow^*_{eval}\exists(\lambda x.\mathbf{boxer}x\wedge\underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z\wedge\mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)\mathsf{c}_b)z)\mathsf{c}_b)}_{\psi}\wedge\\ &\qquad\phi(\mathsf{upd}(\psi,\mathsf{c}_b)))\end{aligned}\tag{6.63}$$

where $\mathsf{c}_b=\mathsf{upd}(\mathbf{boxer}x,\mathsf{c})$.
It is possible to derive $((\lambda y.\mathbf{male}y)x)$ from c_b. Therefore, inner sel in (6.63) returns the individual x, leading to (6.64):

$$\exists(\lambda x.\mathbf{boxer}x\wedge\underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z\wedge\mathbf{poss}xz)\mathsf{c}_b)}_{\psi}\wedge\phi(\mathsf{upd}(\psi,\mathsf{c}_b)))\tag{6.64}$$

However, there is no a such that $\mathsf{c}_b \vdash (\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)\mathsf{a}$. This means that an exception ($\texttt{AbsentIndividualExc}\ (\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)$) is raised. Below it is shown in detail how the exception, abbreviated as ξ, is propagated according to the rules in Definition 6.13:

$\lambda\phi.\exists(\lambda x.\mathbf{boxer}x \wedge \mathbf{esc}x(\mathsf{raise}\ \xi(x)) \wedge \phi(\mathsf{upd}(\mathbf{esc}x(\mathsf{raise}\ \xi(x)), \mathsf{c}_b)))\ \texttt{handle}\ldots$
$\lambda\phi.\exists(\lambda x.\mathbf{boxer}x \wedge (\mathsf{raise}\ \xi(x)) \wedge \phi(\mathsf{upd}((\mathsf{raise}\ \xi(x)), \mathsf{c}_b)))\ \texttt{handle}\ldots$
$\lambda\phi.\exists(\lambda x.(\mathsf{raise}\ \xi(x)) \wedge \phi((\mathsf{raise}\ \xi(x))\mathsf{c}_b))\ \texttt{handle}\ldots$
$\lambda\phi.\exists(\lambda x.(\mathsf{raise}\ \xi(x)) \wedge \phi((\mathsf{raise}\ \xi(x))))\ \texttt{handle}\ldots$
$\lambda\phi.\exists(\lambda x.(\mathsf{raise}\ \xi(x)) \wedge (\mathsf{raise}\ \xi(x)))\ \texttt{handle}\ldots$
$\lambda\phi.\exists(\lambda x.(\mathsf{raise}\ \xi(x)))\ \texttt{handle}\ldots$

The propagation of the exception does not continue as the exception is bound by λ. Therefore, it is not handled by the exception handler in the function dupd_g. □

When a sentence has an indefinite article, the presuppositional content related to the individual introduced by this article has to be accommodated within the scope of the article. To account for this, interpretation (4.38f) (repeated in (6.65)) of the dynamic existential quantifier can be modified by adding an exception handler taking care of this intermediate accommodation.

$$\overline{\exists} \doteq \lambda\mathbf{P}.\lambda e\phi.\exists(\lambda x.\ \mathbf{P}\ (\lambda e'.x)\ e\ \phi) \tag{6.65}$$

The new interpretation $\widetilde{\exists}$ is given in the next definition:

Definition 6.29. Dynamic existential quantifier $\widetilde{\exists}$ is defined as follows:

$$\widetilde{\exists} \doteq \lambda\mathbf{P}.\lambda e\phi.\exists(\lambda x.\ \mathsf{iacc}\ (\mathbf{P}(\lambda e'.x))\ e\ \phi\ x) \tag{6.66}$$

where iacc is a recursive function responsible for intermediate accommodation if the descriptive content of the exception raised is locally bound:

$$\begin{aligned}\mathsf{iacc}\ \mathbf{S}\ e\ \phi\ x \doteq\ & \mathbf{S}\ e\ \phi \\ & \texttt{handle}\ (\texttt{AbsentIndividualExc}\ \mathrm{R})\ \texttt{with} \\ & \quad \texttt{if}\ \ (\mathsf{occurs}\ x\ \mathrm{R})\ \texttt{then}\ \exists(\lambda y.\mathrm{R}y \wedge \mathsf{iacc}\ \ \mathbf{S}\ (\mathsf{upd}(\mathrm{R}y, e))\ \phi\ x) \\ & \quad \texttt{else}\ \ (\mathsf{raise}\ (\texttt{AbsentIndividualExc}\ \mathrm{R}))\end{aligned} \tag{6.67}$$

The dynamic interpretation $\widetilde{[\![a]\!]}$ of the indefinite article that uses $\widetilde{\exists}$ is analogous to the former interpretation $\overline{[\![a]\!]}$:

$$\widetilde{[\![a]\!]} = \lambda\mathbf{PQ}.\widetilde{\exists}(\lambda\mathbf{x}.\mathbf{Px} \barwedge \mathbf{Qx})$$

EXAMPLE 6.30. [$\mathbf{S}'$] The meaning of the noun phrase *a boxer* is as follows:

$$\begin{aligned}
\widetilde{[\![a]\!]}\overline{[\![boxer]\!]} &= (\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px} \mathbin{\overline{\wedge}} \mathbf{Qx})\overline{[\![boxer]\!]} \\
&\rightarrow_{eval} \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.\overline{[\![boxer]\!]}\mathbf{x} \mathbin{\overline{\wedge}} \mathbf{Qx}) \\
&\rightarrow_{eval} \lambda\mathbf{Q}.\overline{\exists}(\lambda\mathbf{x}.\overline{\mathbf{boxer}}\mathbf{x} \mathbin{\overline{\wedge}} \mathbf{Qx}) \\
&\rightarrow^{*}_{eval} \lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda x.\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \mathbf{Q}(\lambda e'.x)e'_b\phi')\ e\ \phi\ x)
\end{aligned}$$

where $e'_b = \mathsf{upd}(\mathbf{boxer}x, e')$.
The meaning of Sentence (56) is then as follows:

$$\begin{aligned}
\mathbf{S}' &= \overline{[\![escaped_from]\!]}(\widetilde{[\!['s]\!]}\widetilde{[\![he]\!]}\overline{[\![apartment]\!]})(\widetilde{[\![a]\!]}\ \overline{[\![boxer]\!]}) \\
&= (\lambda\mathbf{Z}.\mathbf{Z}(\lambda\mathbf{z}.(\lambda e\phi.\underbrace{\mathbf{esc}(\mathbf{z}e)(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e)z)e)}_{\psi} \wedge\phi(\mathsf{upd}(\psi, e)))))(\widetilde{[\![a]\!]}\ \overline{[\![boxer]\!]}) \\
&\rightarrow_{eval} (\widetilde{[\![a]\!]}\ \overline{[\![boxer]\!]})(\lambda\mathbf{z}.(\lambda e\phi.\underbrace{\mathbf{esc}(\mathbf{z}e)(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e)z)e)}_{\psi} \wedge\phi(\mathsf{upd}(\psi, e)))) \\
&= (\lambda\mathbf{Q}.\lambda e\phi.\exists(\lambda x.\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \mathbf{Q}(\lambda e'.x)e'_b\phi')\ e\ \phi\ x)) \\
&\quad (\lambda\mathbf{z}.(\lambda e\phi.\underbrace{\mathbf{esc}(\mathbf{z}e)(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e)z)e)}_{\psi} \wedge\phi(\mathsf{upd}(\psi, e)))) \\
&\rightarrow^{*}_{eval} \lambda e\phi.\exists(\lambda x.\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge\phi'(\mathsf{upd}(\psi, e'_b)))\ e\ \phi\ x)
\end{aligned}$$

□

EXAMPLE 6.31. [$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}'$] Given interpretation $\mathbf{S}'$ of Sentence (56), assume that it is uttered in the same discourse as in Example 6.28 (i.e. in a discourse interpreted as $(\lambda\phi.\phi\mathsf{c})$, where $\mathsf{c} = \forall x.(\mathbf{boxer}x \rightarrow \mathbf{male}x)$).

$$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}' \rightarrow^{*}_{eval} \lambda\phi.\mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}\ \phi \tag{6.68}$$

$$
\begin{aligned}
&\mathbf{S}'\ \mathsf{c}\ \phi \\
\rightarrow^{*}_{eval}\ &(\lambda e\phi.\exists(\lambda x.\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ e\ \phi\ x))\ \mathsf{c}\ \phi \\
\rightarrow^{*}_{eval}\ &\exists(\lambda x.\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ \mathsf{c}\ \phi\ x) \qquad (6.69)
\end{aligned}
$$

The subterm $(\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ \mathsf{c}\ \phi\ x)$ is evaluated as follows. Firstly, $((\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ \mathsf{c}\ \phi)$ is computed:

$$
\begin{aligned}
&(\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ \mathsf{c}\ \phi \\
\rightarrow^{*}_{eval}\ &\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)\mathsf{c}_b)z)\mathsf{c}_b)}_{\psi} \wedge \phi(\mathsf{upd}(\psi, \mathsf{c}_b)) \qquad (6.70)
\end{aligned}
$$

where $\mathsf{c}_b = \mathsf{upd}(\mathbf{boxer}x, \mathsf{c})$. Since $\mathsf{c}_b \vdash (\lambda y.\mathbf{male}y)x$, term $(\mathsf{sel}(\lambda y.\mathbf{male}y)\mathsf{c}_b)$ evaluates to x:

$$
\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)\mathsf{c}_b)}_{\psi} \wedge \phi(\mathsf{upd}(\psi, \mathsf{c}_b)) \qquad (6.71)
$$

However, since there is no a such that $\mathsf{c}_b \vdash (\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)\mathsf{a}$, the selection function sel raises the exception $(\texttt{AbsentIndividualExc}\ (\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz))$. The handler of iacc catches this exception and, since x is locally bound by the existential quantifier (i.e. x occurs in $(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)$), iacc performs the accommodation of the presuppositional content and recursively calls itself with the updated discourse:

$$
\begin{aligned}
&\exists(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz \wedge \qquad (6.72)\\
&\quad \mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ (\mathsf{upd}(\mathbf{ap}z \wedge \mathbf{poss}xz, \mathsf{c}))\ \phi\ x)
\end{aligned}
$$

The evaluation of the subterm
$(\mathsf{iacc}\ (\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ (\mathsf{upd}(\mathbf{ap}z \wedge \mathbf{poss}xz, \mathsf{c}))\ \phi\ x)$ is as follows. Firstly, $((\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ (\mathsf{upd}(\mathbf{ap}z \wedge \mathbf{poss}xz, \mathsf{c}))\ \phi)$ is computed:

$$(\lambda e'\phi'.\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)e'_b)z)e'_b)}_{\psi} \wedge \phi'(\mathsf{upd}(\psi, e'_b)))\ (\mathsf{upd}(\mathbf{ap}z \wedge \mathbf{poss}xz, \mathsf{c}))\ \phi$$
$$\rightarrow^*_{eval} \mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{male}y)\mathsf{c}_{bap})z)\mathsf{c}_{bap})}_{\psi} \wedge \phi(\mathsf{upd}(\psi, \mathsf{c}_{bap}))$$

where $\mathsf{c}_{bap} = \mathsf{upd}(\mathbf{boxer}x, \mathsf{upd}(\mathbf{ap}z \wedge \mathbf{poss}xz, \mathsf{c}))$. Since $\mathsf{c}_{bap} \vdash (\lambda y.\mathbf{male}y)x$, the term $(\mathsf{sel}(\lambda y.\mathbf{male}y)\mathsf{c}_{bap})$ evaluates to x:

$$\mathbf{boxer}x \wedge \underbrace{\mathbf{esc}x(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)\mathsf{c}_{bap})}_{\psi} \wedge \phi(\mathsf{upd}(\psi, \mathsf{c}_{bap}))$$

Moreover, $\mathsf{c}_{bap} \vdash (\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)z$. Therefore, $(\mathsf{sel}(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz)\mathsf{c}_{bap})$ evaluates to z:

$$\mathbf{boxer}x \wedge \mathbf{esc}xz \wedge \phi(\mathsf{upd}(\mathbf{esc}xz, \mathsf{c}_{bap})) \tag{6.73}$$

Composing terms (6.68), (6.69), (6.72) and (6.73), the final interpretation of discourse **D** updated with interpretation $\mathbf{S}'$ of Sentence (56) is obtained:

$$\lambda\phi.\exists(\lambda x.\exists(\lambda z.\mathbf{ap}z \wedge \mathbf{poss}xz \wedge \mathbf{boxer}x \wedge \mathbf{esc}xz \wedge \phi(\mathsf{upd}(\mathbf{esc}xz, \mathsf{upd}(\mathbf{boxer}x, \mathsf{upd}(\mathbf{ap}z \wedge \mathbf{poss}xz, \mathsf{c})))))) \tag{6.74}$$

□

This chapter developed the framework GLχ based on continuation-based dynamic logic with exceptions. On the example of referring expressions it was shown that exception raising can be used to implement the emergence of presuppositions: when the interpretation of a sentence containing a referring expression is evaluated with respect to a context that does not contain an appropriate referent, satisfying the descriptive content P of the referring expression, the exception (`AbsentIndividualExc` P) is raised. Handling of the exception corresponds to accommodation of the presupposition. The chapter suggested possible ways of implementing global and intermediate accommodation and proposed solutions for presupposition projection and binding problems.

GLχ has the potential to be extended even further to express more complex unconventional phenomena and the next chapter informally discusses some possible tracks in this direction.

Chapter 7

Directions for Further Development of the Framework

Due to its simplicity and flexibility, framework GLχ has potential to be extended for handling even more complex phenomena in natural language than those described in Chapters 5 and 6. This chapter informally discusses some directions for possible extensions of the framework. Section 7.1 illustrates how presuppositions triggered by the verb *know* can be handled in GLχ. Section 7.2 outlines an approach to deal with some conversational implicatures. Section 7.3 concludes by making a comparison of presuppositions and conversational implicatures using the terminology of GLχ. Thereby, this chapter informally illustrates the possibility to handle semantic and (some) pragmatic phenomena in a unified framework.

7.1 Presuppositions

Chapter 5 presented how presuppositions stemming from referring expressions can be expressed in GLχ and Chapter 6 showed how GLχ deals with the issues related to presupposition triggering. Among these issues is finding out whether the presupposition has to be accommodated or not and at which place in the logical formula the presupposition projects. To account these issues GLχ uses continuation passing technique and exception raising and handling mechanism. These techniques are general and can be applied to interpret other presuppositional phenomena. Section 7.1.1 proposes a solution for factive verbs on the example of *know*.

7.1.1 Factive Verbs

Presuppositions can be triggered by factive verbs. For example, when the verb *to know* takes a noun clause as its direct object, the resulting verb phrase becomes a presupposition triggering expression. Consider, for example, Sentence (57):

(57) *Tom knows that John loves Mary.*

If a hearer is not aware that John loves Mary, the content of the subordinate clause becomes for him/her as prominent as the content of the main sentence. In other words, the hearer accommodates the meaning conveyed in the subordinate clause into his/her knowledge base (context). However, if the hearer already knows that John loves Mary, his/her knowledge is updated with the content of the main clause only. Below it is shown on example (57) how this kind of presupposition can be handled with the framework based on continuation-based dynamic logic with exceptions.

According to the syntactic parse tree of the sentence, shown in Figure 7.1, its meaning has to be computed by β-reducing term (7.1):

$$\overline{[\![knows]\!]}(\overline{[\![that]\!]}(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}))\widetilde{[\![Tom]\!]} \tag{7.1}$$

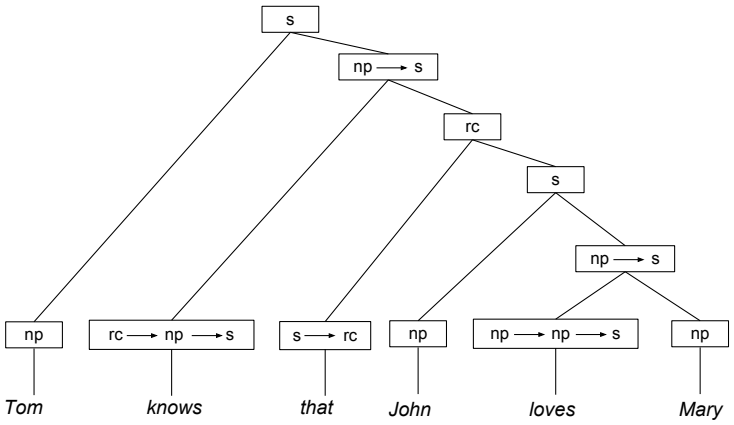


Figure 7.1: Syntactic parse tree of sentence *Tom knows that John loves Mary.*

Before computing the meaning of the sentence, a remark related to the type of the factive verb *know* is beneficial. Standard (type-raised) interpretations of transitive verbs take two (type-raised) noun phrases as arguments and return a proposition. For example, a familiar static interpretation of *knows* is presented in (7.2), where **know** is a non-logical constant of type $(\iota \to \iota \to o)$:

$$[\![knows]\!] = \lambda \mathrm{YX.X}(\lambda x.\mathrm{Y}(\lambda y.\mathbf{know}xy)) \tag{7.2}$$

However, the direct object of the verb in (57) is not a noun phrase, but a subordinate clause. Compare two sentences in (58). The verb *knows* in (58a) has a type different from the type of the verb *knows* in (58b). In (58a) *knows* takes a fact as its direct object. In (58b) *knows* takes an individual as its direct object. In fact, in some languages, for example, in French (59), Portuguese (60) and German (61), different verbs are used in these two different cases.

(58) a. *Tom knows that John loves Mary.*

b. *Tom knows John.*

(59) a. *Tom sait que John aime Mary.*

b. *Tom connaît John.*

(60) a. *Tom sabe que John ama Mary.*

b. *Tom conhece John.*

(61) a. *Tom weiß, dass John Mary liebt.*

b. *Tom kennt John.*

A possible way to handle this for English is to add an alternative interpretation for the factive verb *know*. This alternative interpretation corresponds to the interpretation of *savoir* in French and has a different type: it takes a proposition instead of an individual as one of its arguments. Therefore, the interpretation of *knows* can be defined as a function of type $(o \to ((\iota \to o) \to o) \to o)$ and, therefore, its dynamic version $\overline{[\![knows]\!]}$ as a function of type $(\overline{o} \to ((\overline{\iota} \to \overline{o}) \to \overline{o}) \to \overline{o})$. Equation (7.3) presents such a λ-term for $\overline{[\![knows]\!]}$:

$$\overline{[\![knows]\!]} = \lambda \mathbf{QX}.\mathbf{X}(\lambda \mathbf{x}.\overline{\mathbf{know}}\mathbf{xQ}) \tag{7.3}$$

The constant **know** in (7.3) is of type $(\iota \to o \to o)$. Its dynamic version $\overline{\mathbf{know}}$ (of type $(\overline{\iota} \to \overline{o} \to \overline{o})$) is computed below in accordance with the dynamization rules defined in 4.27:

$$\begin{aligned}
\overline{\mathbf{know}} &= \mathbb{D}_{\iota \to o \to o}[\lambda e.\mathbf{know}] && \text{(by (4.38a))}\\
&= \lambda \mathbf{a}.\mathbb{D}_{o \to o}[\lambda e.\mathbf{know}\ \mathbb{S}_{\iota}[\mathbf{a}, e]] && \text{(by (4.36c))}\\
&= \lambda \mathbf{a}.\mathbb{D}_{o \to o}[\lambda e.\mathbf{know}(\mathbf{a}e)] && \text{(by (4.37a))}\\
&= \lambda \mathbf{aA}.\mathbb{D}_{o}[\lambda e.\mathbf{know}(\mathbf{a}e)\mathbb{S}_{o}[\mathbf{A}, e]] && \text{(by (4.36c))}\\
&= \lambda \mathbf{aA}.\mathbb{D}_{o}[\lambda e.\mathbf{know}(\mathbf{a}e)(\mathbf{A}e(\lambda e.\top))] && \text{(by (4.37b))}\\
&= \lambda \mathbf{aA}.\lambda e\phi.\mathbf{know}(\mathbf{a}e)(\mathbf{A}e(\lambda e.\top)) \wedge \phi(\mathsf{upd}(\mathbf{know}(\mathbf{a}e)(\mathbf{A}e(\lambda e.\top))), e) && (7.4)
\end{aligned}$$

To express the presupposition triggering effect, (7.4) should be modified. The change takes place in the second argument of $\overline{\mathbf{know}}$ standing for the relative clause. Particularly, the function $\mathsf{checkvalid}$ of type $(o \to \gamma \to o)$ is added, as shown in (7.5):

$$\begin{aligned}
\widetilde{\mathbf{know}} = \lambda \mathbf{aA}.\lambda e\phi.\mathbf{know}(\mathbf{a}e)(\mathsf{checkvalid}\ (\mathbf{A}e(\lambda e.\top))\ e) \wedge \\
\phi(\mathsf{upd}(\mathbf{know}(\mathbf{a}e)(\mathsf{checkvalid}\ (\mathbf{A}e(\lambda e.\top))\ e), e))
\end{aligned} \tag{7.5}$$

Function checkvalid takes a proposition p and an environment as arguments and tries to prove this proposition based on the knowledge contained in the environment. If it succeeds, it returns the proposition unchanged. If it fails, it triggers an exception:

Definition 7.1. Let p be a term of type ι and c be a term of type o. Then $\mathsf{checkvalid}\ p\ c \equiv p$ is defined as follows:

$$\mathsf{checkvalid}\ p\ c = \begin{cases} p & \text{if } c \vdash p \\ \mathsf{raise}\ (\texttt{UnprovablePropExc}\ p) & \text{otherwise} \end{cases}$$

Then, using interpretation (7.5) of the constant $\widetilde{\mathbf{know}}$ in (7.3), the normalized dynamic interpretation of *knows* is as follows:

$$\begin{aligned} & \widetilde{[\![knows]\!]} \\ &= \lambda\mathbf{Q}\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\lambda e\phi.\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{Q}e(\lambda e.\top))e)\wedge \\ &\qquad\qquad \phi(\mathsf{upd}(\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{Q}e(\lambda e.\top))e), e))) \\ &= \lambda\mathbf{Q}\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{Q}e(\lambda e.\top))e)}_{\psi}\wedge\phi\ (\mathsf{upd}(\psi, e))) \end{aligned} \tag{7.6}$$

The remaining dynamic interpretation for *that* is the term of type $(\overline{o} \to \overline{o})$:

$$\overline{[\![that]\!]} = \lambda\mathbf{S}.\mathbf{S} \tag{7.7}$$

Next example computes the meaning of Sentence (57) using interpretation (7.6) for the factive verb and interpretation (7.7) for the relative pronoun. Dynamic interpretations of other lexical items are as usual.

EXAMPLE 7.2. [*Tom knows that John loves Mary*] The dynamic meaning of *John loves Mary* is computed in Example 5.9 and repeated below in (7.8). The normal form of (7.8) is (7.9):

$$\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]} \to^*_{eval} \overline{\mathbf{love}}(\mathsf{sel}(\text{named "John"}))(\mathsf{sel}(\text{named "Mary"})) \tag{7.8}$$

$$\to^*_{eval} \lambda e\phi.\underbrace{\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi, e)) \tag{7.9}$$

Then, the meaning of *that John loves Mary* is as follows:

$$\begin{aligned}
\overline{[\![that]\!]}(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}) &= (\lambda\mathbf{S}.\mathbf{S})(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}) && \text{(by (7.7))}\\
&\to_{eval} \overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}\\
&= \lambda e\phi.\underbrace{\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi, e)) && \text{(by (7.9))}
\end{aligned}$$

Now the meaning of *knows that John loves Mary* can be computed:

$$\begin{aligned}
&\widetilde{[\![knows]\!]}(\overline{[\![that]\!]}(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}))\\
&= (\lambda\mathbf{Q}\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{Q}e(\lambda e.\top))e)}_{\psi} \wedge \phi(\mathsf{upd}(\psi, e))))(\overline{[\![that]\!]}(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}))\\
&\to_{eval} \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\overline{[\![that]\!]}(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]})e(\lambda e.\top))e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi, e)))\\
&\to^*_{eval} \lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e))e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi, e)))
\end{aligned} \tag{7.10}$$

Finally, the normalized meaning **S** (7.11) of the sentence is computed in the following way:

$$
\begin{aligned}
\mathbf{S} &= \widetilde{[\![know]\!]}(\overline{[\![that]\!]}(\overline{[\![loves]\!]}\ \widetilde{[\![Mary]\!]}\ \widetilde{[\![John]\!]}))\widetilde{[\![Tom]\!]} \\
&= (\lambda\mathbf{X}.\mathbf{X}(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\text{named ``Mary''})e))e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi,e))))\widetilde{[\![Tom]\!]} \\
\rightarrow_{eval}\ &\widetilde{[\![Tom]\!]}(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\text{named ``Mary''})e))e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi,e))) \\
&= (\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\text{named ``Tom''})))(\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\text{named ``Mary''})e))e)}_{\psi} \wedge \\
&\qquad \phi\ (\mathsf{upd}(\psi,e))) \\
&= (\lambda\mathbf{x}.\lambda e\phi.\underbrace{\mathbf{know}(\mathbf{x}e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\text{named ``Mary''})e))e)}_{\psi} \wedge \phi(\psi,e)))(\mathsf{sel}(\text{named ``Tom''})) \\
&= \lambda e\phi.\underbrace{\mathbf{know}(\mathsf{sel}(\text{named ``Tom''})e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named ``John''})e)(\mathsf{sel}(\text{named ``Mary''})e))e)}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi,e))
\end{aligned}
\tag{7.11}
$$

□

7.1.2 Additional exception handler in discourse update

As discussed in Section 6.1, exceptions are raised and handled on the level of discourse, i.e. when a sentence is added to a preceding discourse. To handle the new kind of exception related to the side effects of factive verbs (i.e. exceptions raised by checkvalid as defined in 7.1), the relevant handler should be added to the discourse update function defined in 6.6. Hence, in addition to the handler (7.12c) for the exceptions triggered by the function sel, the discourse update function contains now the handler (7.12d), for exceptions of the form (`UnprovablePropExc` P) raised by the function checkvalid:

Definition 7.3. [Discourse update] Let $\mathbf{D}$ be an interpretation of a discourse and $\mathbf{S}$ be an interpretation of a sentence. Then the function $\mathsf{dupd}_{\mathsf{g}}$ is defined by Equation (7.12a), where gacc is a function of type $((\gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o)$ that is responsible for global accommodation and is defined in (7.12b)–(7.12d):

$$\mathsf{dupd}_{\mathsf{g}}\ \mathbf{D}\ \mathbf{S} \doteq \lambda\phi.\ \mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \tag{7.12a}$$

$$\mathsf{gacc}\ \mathbf{S}\ e\ \phi \doteq \mathbf{S}\ e\ \phi \tag{7.12b}$$

$$\texttt{handle}\ (\texttt{AbsentIndividualExc}\ \mathrm{Q})\ \texttt{with}$$
$$\exists(\lambda x.(\mathrm{Q}x) \wedge \mathsf{gacc}\ \mathbf{S}\ (\mathsf{upd}(\mathrm{Q}x, e))\ \phi) \tag{7.12c}$$

$$\texttt{handle}\ (\texttt{UnprovablePropExc}\ \mathrm{P})\ \texttt{with}$$
$$\mathsf{gacc}\ \mathbf{S}\ (\mathsf{upd}(\mathrm{P}, e))\ \phi \tag{7.12d}$$

Exception (`UnprovablePropExc` P) contains a proposition P that could not be proved in the knowledge base of the context. Note that this proposition is only added to the context in (7.12d). The assertion (content) remains unchanged. This goes in line with Stalnaker's [1974] idea of separating content and context:

> First remark: semantics, as contrasted with pragmatics, can mean either the study of *meaning* or the study of *context*. [...] Second remark: in recommending a separation of content and context I am not suggesting that there is no interaction between them. Far from it. The semantic rules which determine the content of a sentence may do so only relative to the context in which it is uttered. [Stalnaker, 1974]

The interaction between content and context is accomplished here via the exception raising mechanism: an exception is raised if a particular content contained in the sentence cannot be selected or proved in the context of the discourse. This is demonstrated in more details in the next example.

EXAMPLE 7.4. Assume that *Tom knows that John loves Mary* is uttered in a discourse which meaning $\mathbf{D}$ is defined in (7.13), where context c_{tjm} contains only knowledge that there exist individuals named "Tom", "John" and "Mary", as shown in (7.14):

$$\mathbf{D} = \lambda\phi.\exists(\lambda t.(\text{named "Tom"}\,t) \wedge \exists(\lambda j.(\text{named "John"}\,j) \wedge \exists(\lambda m.(\text{named "Mary"}\,m) \wedge \phi\mathsf{c}_{tjm}))) \tag{7.13}$$

$$\mathsf{c}_{tjm} = \mathsf{upd}(\text{named "Mary"}\,m, \mathsf{upd}(\text{named "John"}\,j, \mathsf{upd}(\text{named "Tom"}\,t, \top))) \tag{7.14}$$

Below the meaning $\mathbf{D}$ updated with $\mathbf{S}$ is computed in accordance with Definition (7.12):

$$\begin{aligned}
&\mathsf{dupd}_{\mathsf{g}}\ \mathbf{D}\ \mathbf{S}\\
&= \lambda\phi.\ \mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) && \text{(by (7.12a))}\\
&= \lambda\phi.(\lambda\phi.\exists(\lambda t.(\text{named "Tom"}\,t) \wedge \exists(\lambda j.(\text{named "John"}\,j) \wedge \exists(\lambda m.(\text{named "Mary"}\,m) \wedge \phi\mathsf{c}_{tjm}))))(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\\
&\to_{eval} \lambda\phi.\exists(\lambda t.(\text{named "Tom"}\,t) \wedge \exists(\lambda j.(\text{named "John"}\,j) \wedge \exists(\lambda m.(\text{named "Mary"}\,m) \wedge (\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\mathsf{c}_{tjm})))\\
&\to_{eval} \lambda\phi.\exists(\lambda t.(\text{named "Tom"}\,t) \wedge \exists(\lambda j.(\text{named "John"}\,j) \wedge \exists(\lambda m.(\text{named "Mary"}\,m) \wedge \mathsf{gacc}\ \mathbf{S}\ \mathsf{c}_{tjm}\ \phi))) && (7.15)
\end{aligned}$$

The computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}_{tjm}\ \phi)$ of (7.15):

$$\begin{aligned}
&\mathbf{S}\ \mathsf{c}_{tjm}\ \phi\\
&= (\lambda e\phi.\underbrace{\mathbf{know}(\mathsf{sel}(\text{named "Tom"})e)(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named "John"})e)(\mathsf{sel}(\text{named "Mary"})e))e)}_{\psi} \wedge\\
&\qquad \phi(\mathsf{upd}(\psi, e)))\mathsf{c}_{tjm}\phi\\
&\to^{*}_{eval} \underbrace{\mathbf{know}(\mathsf{sel}(\text{named "Tom"})\mathsf{c}_{tjm})(\mathsf{checkvalid}(\mathbf{love}(\mathsf{sel}(\text{named "John"})\mathsf{c}_{tjm})(\mathsf{sel}(\text{named "Mary"})\mathsf{c}_{tjm}))\mathsf{c}_{tjm})}_{\psi} \wedge\\
&\qquad \phi(\mathsf{upd}(\psi, \mathsf{c}_{tjm})) && (7.16)
\end{aligned}$$

Since the environment c_{tjm} contains knowledge about the existence of individuals Tom, John and Mary, the selection function sel in (7.16) successfully returns corresponding variables resulting in (7.17):

$$\underbrace{\mathbf{know}\ t\ (\mathsf{checkvalid}\ (\mathbf{love}jm)\mathsf{c}_{tjm})}_{\psi} \wedge \phi(\mathsf{upd}(\psi, \mathsf{c}_{tjm})) \tag{7.17}$$

However, the function checkvalid raises an exception in (7.17). This is caused by the fact that the proposition $\mathbf{love}jm$ cannot be proved based on the knowledge contained in the environment c_{tjm}. The exception is handled according to (7.12d). Therefore, the term (gacc **S** ($\mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{tjm})$) ϕ) is evaluated next leading to (7.18):

$$\underbrace{\mathbf{know}\ t\ (\mathsf{checkvalid}(\mathbf{love}jm)\ (\mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{tjm})))}_{\psi} \wedge \phi\ (\mathsf{upd}(\psi, \mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{tjm}))) \tag{7.18}$$

This time the function checkvalid is able to prove $\mathbf{love}jm$ based on the knowledge contained in the updated environment $\mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{tjm})$ and it returns the proposition. This leads to (7.19):

$$\mathbf{know}\ t\ (\mathbf{love}jm) \wedge \phi(\mathsf{upd}(\mathbf{know}\ t\ (\mathbf{love}jm), \mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{tjm}))) \tag{7.19}$$

Then, combining (7.15) with (7.19), the resulting meaning of the updated discourse is as follows:

$$\begin{aligned}\lambda\phi.\exists(\lambda t.(\text{named "Tom"}t) \wedge \exists(\lambda j.(\text{named "John"}j) \wedge \exists(\lambda m.(\text{named "Mary"}m) \wedge (\mathbf{know}\ t\ (\mathbf{love}jm) \wedge \\ \phi(\mathsf{upd}(\mathbf{know}\ t(\mathbf{love}jm), \mathsf{upd}(\mathbf{love}jm, \mathsf{c}_{tjm})))))))\end{aligned} \tag{7.20}$$

□

7.2 Conversational Implicatures

The framework presented in this dissertation can be extended to handle other kinds of linguistic side effects such as conversational implicatures [Grice, 1975], which are informally what is suggested by an utterance but neither explicitly said nor entailed by the utterance.

Chapter 5 and in Section 7.1 showed, respectively, how presuppositions stemming from referring expressions and factive verbs can be handled in framework GLχ having an exception raising and handling mechanism. Triggering of a presupposition requires a search for a proof of the required property in the context. If the search fails, an exception is raised. The code that handles the exception corresponds to accommodation of the presupposition. This section elaborates [Lebedeva and Woltzenlogel Paleo, 2010] and shows that a similar exception-based approach can, at least in principle, be used for computing conversational implicatures.

7.2.1 Conversational Implicatures by Proof-Theoretical Abduction

Although evidence of the study of abductive reasoning can be already found in the work of Aristotle [350 BCE], Pierce is considered to be the first who explored it broadly [Hobbs, 2004]. In [1955, p.151] Pierce characterizes abduction in the following way:

> [...] the operation of adopting an explanatory hypothesis - which is just what abduction is - was subject to certain conditions. Namely, the hypothesis cannot be admitted, even as a hypothesis, unless it be supposed that it would account for the facts or some of them. The form of inference, therefore, is this:
>
> The surprising fact, C, is observed;
> But if A were true, C would be a matter of course,
> Hence, there is reason to suspect that A is true.

More formally, the abduction problem can be defined as follows:

Definition 7.5. [Abduction problem] Let K be a theory (i.e. a set of formulas). Let F be a formula such that $K \nvDash F$. The **abduction problem** is concerned with finding a set of statements S such that $K \cup S \vDash F$.

Note that, in unrestricted form as defined in 7.5, the problem is trivial: it is sufficient to take S equal to $\{F\}$. Therefore, usually some kind of minimality is required for the set S.

The explanation of certain natural language phenomena via abductive reasoning has already been proposed in [Hobbs *et al.*, 1993]. One of the most intriguing applications of this approach is computing conversational implicatures. Indeed, conversational implicatures are related to abductive reasoning in the sense that "an implicature can be viewed as an abductive move for the sake of achieving the best interpretation" [Hobbs, 2004].[1]

The abductive approach of computing implicatures involves trying to prove the interpretation (logical form) of an uttered sentence in the current knowledge base:

> By [...] [prove the logical form of the sentence] we mean "prove, or derive in the logical sense, from the predicate calculus axioms in the knowledge base, the logical form that has been produced by syntactic analysis and semantic translation of the sentence." [Hobbs, 2004]

The failure to find a proof requires abduction, which adds to the knowledge base facts necessary for the proof. These facts are exactly the implicatures of the sentence in the original knowledge base.

Adopting this scheme to a dialogue, imagine a sentence S was uttered by A. The hearer B tries to prove S from his/her knowledge and in the case of failure, B abduces the implicatures of S and accommodates them as new knowledge. Hence, with respect to a dialogue, the knowledge base can be understood as the knowledge of the hearer of an uttered sentence.

Hobbs *et al.* [1993] develop their abductive approach under the assumption that the logical form of the sentence and the knowledge base are given. The framework developed in this dissertation can be extended to incorporate an abductive reasoning system. Therefore, the resulting framework would not only have an abduction mechanism, but also be able to apply it compositionally as it has means for compositional computation of logical forms of sentences and update of the knowledge base.

Informally, the abductive approach can be integrated in frameworg GLχ in the following way. Consider the context c as a representation of the knowledge base (assuming that context is defined as a set of formulas). During the processing of the discourse, a proof of the current sentence from the axioms contained in the context is performed. If the search fails, an exception is raised. Handling the exception consists of abducing facts that are needed to complete the failed proofs. The abduced facts are the implicatures of the sentence, they are computed from the failed proofs and added to the context c.

To see how it works on an example, first imagine that the participant B of the future dialogue knows the following statements (she/he could have acquired it based on personal experience or previous conversations).

[1] Hobbs *et al.* [1993] use a technique called weighted abduction as a criteria for determining the best proof. In brief, they assign certain weights to propositions and the goal is to construct a proof with minimal weight.

- Everybody dates only one person.
- Dating implies going out.
- Jane dates Vincent.
- Vincent and Tom are different persons.

The corresponding knowledge base (i.e. context) is as follows:

- $\forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y))$
- $\forall x \forall y.(D(x,y) \rightarrow G(x,y))$
- $D(j,v)$
- $v \neq t$

Now, keeping the knowledge base of B in mind, consider that A says the following:

A: Jane went out with Tom yesterday night.

What happens in B's mind can be modelled according to the following scheme. First of all, B checks if A's statement is entailed by B's knowledge base. This requires a search for a proof of the logical interpretation $G(j,t)$ of A's sentence from the formulas contained in B's context:[2]

$$v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), \forall x \forall y.(D(x,y) \rightarrow G(x,y)) \vdash G(j,t)$$

B realizes that A's statement cannot be entailed from B's knowledge because B fails to find a proof. Indeed, B cannot complete the proof search and ends up with partial proofs as shown below:

$$\dfrac{\dfrac{\dfrac{\vdash D(j,t) \qquad \overline{G(j,t) \vdash G(j,t)}}{D(j,t) \rightarrow G(j,t) \vdash G(j,t)} \rightarrow_l}{\forall x \forall y.(D(x,y) \rightarrow G(x,y)) \vdash G(j,t)} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), \forall x \forall y.(D(x,y) \rightarrow G(x,y)) \vdash G(j,t)} w_l$$

The proof search can be defined in a way that it raises an exception `UnprovablePropExc` if the proof is not found (analogously to exceptions raised when the selection function sel cannot find a required individual in the context (Chapter 5) and the checking-validity function checkvalid fails to prove a proposition in the context (Section 7.1)). The abduction of the needed facts should be performed, then, by the handler of `UnprovablePropExc`. In the current example B concludes (abduces), based on his/her knowledge, that Jane dates Tom:

[2] The framework is independent of the proof calculus used, in the same way as it is independent of the anaphora resolution algorithm in sel. Here and further on the sequent calculus is used due to its convenience.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{D(j,t) \vdash D(j,t)} \quad \dfrac{}{G(j,t) \vdash G(j,t)}}{D(j,t) \to G(j,t) \vdash G(j,t)} \to_l}{\forall x \forall y.(D(x,y) \to G(x,y)) \vdash G(j,t)} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \to \neg D(z,y)), \forall x \forall y.(D(x,y) \to G(x,y)) \vdash G(j,t)} w_l$$

Fixing the proof requires the downward propagation of the abduced facts:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{D(j,t) \vdash D(j,t)} \quad \dfrac{}{G(j,t) \vdash G(j,t)}}{D(j,t), D(j,t) \to G(j,t) \vdash G(j,t)} \to_l}{D(j,t), \forall x \forall y.(D(x,y) \to G(x,y)) \vdash G(j,t)} \forall_l}{D(j,t), v \neq t, D(j,v), W, \forall x \forall y.(D(x,y) \to G(x,y)) \vdash G(j,t)} w_l$$

where W is $\forall x \forall y \forall z.(x \neq y \wedge D(z,x) \to \neg D(z,y))$. This corresponds to adding the abduced knowledge to B's knowledge base:

- Everybody dates only one person: $\forall x \forall y \forall z.(x \neq y \wedge D(z,x) \to \neg D(z,y))$
- Dating implies going out: $\forall x \forall y.(D(x,y) \to G(x,y))$
- Jane dates Vincent: $D(j,v)$
- Vincent and Tom are different persons: $v \neq t$
- Jane dates Tom: $D(j,t)$

However, updating the context with the abduced fact leads, in this case, to a unsatisfiable context. The next section discusses this issue in detail.

7.2.2 Conversational Implicatures by Correcting the Context

This subsection continues with pragmatical investigation of how correcting the context, after adding the abduced fact that Jane dates Tom, can cause new conversational implicatures. This can result in different responses of B to A's statement that Jane went out with Tom yesterday night.

Adding the abduced fact to B's knowledge base, resulted in an unsatisfiable knowledge base. A sequent calculus refutation is shown below:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v \neq t \vdash v \neq t} \quad \dfrac{}{D(j,v) \vdash D(j,v)}}{v \neq t, D(j,v) \vdash t \neq v, D(j,v)} \wedge_r \quad \dfrac{\dfrac{}{D(j,t) \vdash D(j,t)}}{\neg D(j,t), D(j,t) \vdash} \neg_l}{v \neq t, D(j,v), (t \neq v \wedge D(j,v) \to \neg D(j,t)), D(j,t) \vdash} \to_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \to \neg D(z,y)), D(j,t) \vdash} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \to \neg D(z,y)), \forall x \forall y.(D(x,y) \to G(x,y)), D(j,t) \vdash} w_l$$

Analyzing this proof, it is possible to see that the following facts are necessary for the refutation:

$$v \neq t$$
$$D(j, v)$$
$$\forall x \forall y \forall z.(x \neq y \wedge D(z, x) \rightarrow \neg D(z, y))$$
$$D(j, t)$$

Therefore, removal of (at least) one of these facts makes the knowledge base consistent again.

This can be implemented in GLχ by defining a function `con` that checks consistensy of a knowledge base and raises exception `InconsistentContextExc` if the knowledge base is inconsistent. The handler of `InconsistentContextExc` should perform the analysis of the proof to make the knowledge base consistent.

B has a choice of which fact can be eliminated. By choosing one of them, B might need to confirm with A if what he/she is going to reject in his/her current knowledge indeed does not hold (according to A) any more. B does it by responding to A's utterance with a sentence, which meaning is the negated fact to be removed.

Assume, for example, that B chooses to reconsider B's knowledge about Jane dating Vincent, i.e. B decides to remove the fact $D(j, v)$ from the knowledge base:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v \neq t \vdash v \neq t} \quad \dfrac{}{D(j,v) \vdash D(j,v)}}{v \neq t, D(j,v) \vdash t \neq v, D(j,v)} \wedge_r \quad \dfrac{\dfrac{}{D(j,t) \vdash D(j,t)}}{\neg D(j,t), D(j,t) \vdash} \neg_l}{v \neq t, D(j,v), (t \neq v \wedge D(j,v) \rightarrow \neg D(j,t)), D(j,t) \vdash} \rightarrow_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), D(j,t) \vdash} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), \forall x \forall y.(D(x,y) \rightarrow G(x,y)), D(j,t) \vdash} w_l$$

This corresponds to the conversational implicature that Jane does not date Vincent. B reacts by exclaiming his surprise:

A: Jane went out with Tom yesterday night.
B: Ah! So, Jane is not dating Vincent anymore!

Another option for B is to reject the fact that everybody dates only one person $\forall x \forall y \forall z.(x \neq y \wedge D(z, x) \rightarrow \neg D(z, y))$ or, its instantiation that Jane dates only one person $(t \neq v \wedge D(j, v) \rightarrow \neg D(j, t))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v \neq t \vdash v \neq t} \quad \dfrac{}{D(j,v) \vdash D(j,v)}}{v \neq t, D(j,v) \vdash t \neq v, D(j,v)} \wedge_r \quad \dfrac{\dfrac{}{D(j,t) \vdash D(j,t)}}{\neg D(j,t), D(j,t) \vdash} \neg_l}{v \neq t, D(j,v), (t \neq v \wedge D(j,v) \rightarrow \neg D(j,t)), D(j,t) \vdash} \rightarrow_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), D(j,t) \vdash} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), \forall x \forall y.(D(x,y) \rightarrow G(x,y)), D(j,t) \vdash} w_l$$

B's response can, therefore, be something like the following:

A: Jane went out with Tom yesterday night.
B: Jane is cheating on Vincent! Uhhh!

B can also consider "Vincent" and "Tom" as names of the same person, and thus remove the fact $v \neq t$ from his or her knowledge base:[3]

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v \neq t \vdash v \neq t} \quad \dfrac{}{D(j,v) \vdash D(j,v)}}{v \neq t, D(j,v) \vdash t \neq v, D(j,v)} \wedge_r \quad \dfrac{\dfrac{}{D(j,t) \vdash D(j,t)}}{\neg D(j,t), D(j,t) \vdash} \neg_l}{v \neq t, D(j,v), (t \neq v \wedge D(j,v) \rightarrow \neg D(j,t)), D(j,t) \vdash} \rightarrow_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), D(j,t) \vdash} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), \forall x \forall y.(D(x,y) \rightarrow G(x,y)), D(j,t) \vdash} w_l$$

Then B can continue the dialogue by telling A that B is already aware that Jane and Tom/Vincent are dating:[4]

A: Jane went out with Tom yesterday night.
B: Ah ya! Indeed, she is dating Vincent.

The final possibility for B is not to accept the abduced fact $D(j,t)$, i.e. reject the implicature that Jane dates Tom:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v \neq t \vdash v \neq t} \quad \dfrac{}{D(j,v) \vdash D(j,v)}}{v \neq t, D(j,v) \vdash t \neq v, D(j,v)} \wedge_r \quad \dfrac{\dfrac{}{D(j,t) \vdash D(j,t)}}{\neg D(j,t), D(j,t) \vdash} \neg_l}{v \neq t, D(j,v), (t \neq v \wedge D(j,v) \rightarrow \neg D(j,t)), D(j,t) \vdash} \rightarrow_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), D(j,t) \vdash} \forall_l}{v \neq t, D(j,v), \forall x \forall y \forall z.(x \neq y \wedge D(z,x) \rightarrow \neg D(z,y)), \forall x \forall y.(D(x,y) \rightarrow G(x,y)), D(j,t) \vdash} w_l$$

Then B can let A know that A is probably wrong:

A: Jane went out with Tom yesterday night.
B: No! It can't be. Jane is dating Vincent and she is a faithful person.

The choice of the fact(s) to be removed should also be made in the handler of `InconsistentContextExc`.

[3] This is a common case in Russian, for example, where almost every name has numerous equivalent variants.

[4] This version of the dialogue appears to be strange because, in fact, "Tom" and "Vincent" are different names. Imagine a similar dialogue with names "Alexander" and "Sasha" happens in Russia (where these names are synonymous):

A: Jane went out with Alex yesterday night.
B: Ah ya! Indeed, she is dating Sasha.

7.2.3 Additional exception handlers in discourse update

Based on the informal analysis of computing conversational implicatures in two previous subsections, a new version of the discourse update function $\mathsf{dupd_g}$ can be defined. In particular, to handle exception `UnprovablePropExc` raised by the failure to find the proof of an asserted statement (having logical formula F), an additional handler (7.21d) that fixes the proof by abducing necessary facts should be added. Moreover, to handle exception `InconsistentContextExc` caused by updating the context e with a (abduced) fact, another handler (7.21e) that fixes the context by removing axioms necessary for the refutation of e should be added. Thus, the definition is as follows:

Definition 7.6. [Discourse update] Let $\mathbf{D}$ be an interpretation of a discourse and $\mathbf{S}$ be an interpretation of a sentence. Then the function $\mathsf{dupd_g}$ is defined by Equation (7.21a), where gacc is a function of type $((\gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o)$ that is responsible for global accommodation and is defined in (7.21b)–(7.21e):

$$\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S} \doteq \lambda\phi.\ \mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) \tag{7.21a}$$

$$\mathsf{gacc}\ \mathbf{S}\ e\ \phi \doteq \mathsf{checkprovable}\ \mathbf{S}\ e\ \phi \tag{7.21b}$$

$$\texttt{handle}\ (\texttt{AbsentIndividualExc}\ \mathrm{Q})\ \texttt{with}$$

$$\exists(\lambda x.(\mathrm{Q}x) \wedge \mathsf{gacc}\ \mathbf{S}\ (\mathsf{upd}(\mathrm{Q}x, e))\ \phi) \tag{7.21c}$$

$$\texttt{handle}\ (\texttt{UnprovablePropExc}\ \mathrm{F})\ \texttt{with}$$

$$\mathsf{gacc}\ \mathbf{S}\ (\mathsf{con}(\mathsf{abd}(\mathrm{F}, e)))\ \phi \tag{7.21d}$$

$$\texttt{handle}\ (\texttt{InconsistentContextExc}\ e')\ \texttt{with}$$

$$\mathsf{gacc}\ \mathbf{S}\ (\mathsf{makesat}\ e')\ \phi \tag{7.21e}$$

One of the novelties of $\mathsf{dupd_g}$ above is the existence of the function $\mathsf{checkprovable}$, which is the very first function called in the body of gacc, as shown in (7.21b). This function, more formally defined below, can raise `UnprovablePropExc` when a proposition cannot be proven from a context:

Definition 7.7. Let $\mathbf{S}$ be an interpretation of a sentence, e be a context of type o and ϕ be a term of type $(\gamma \to o)$. Then $(\mathsf{checkprovable}\ \mathbf{S}\ e\ \phi)$ is defined as follows:

$$\mathsf{checkprovable}\ \mathbf{S}\ e\ \phi \doteq \begin{cases} \mathbf{S}\ e\ \phi & \text{if } e \vdash \mathbf{S}e(\lambda e.\top) \\ \mathsf{raise}\ (\texttt{UnprovablePropExc}\ (\mathbf{S}e(\lambda e.\top))) & \text{otherwise} \end{cases}$$

Thus, by raising `UnprovablePropExc`, the function $\mathsf{checkprovable}$ signals that the stated proposition cannot be proven from the context of the previous discourse. The exception is handled in (7.21d) by abduction. Function abd takes a formula F and a context e, constructs a proof of F from the axioms in e and returns e updated with abduced facts. Abduced facts that are not equal to $\mathbf{S}e(\lambda e.\top)$ are the conversational implicatures.

Function con checks whether abduced facts cause inconsistency of the context and raises exception `InconsistentContextExc` if this is the case:

Definition 7.8. Let e be a context of type o. Then con e is defined as follows:

$$\mathsf{con}\ e = \begin{cases} e & \text{if } \mathsf{consistent}\ e \\ \mathsf{raise}\ (\texttt{InconsistentContextExc}\ e) & \text{otherwise} \end{cases}$$

Exception `InconsistentContextExc` is handled in (7.21e) with the function makesat. This function makes the unsatisfiable context e satisfiable by removing facts from it and returns the obtained satisfiable context.

Note that the exception `UnprovablePropExc` is handled in (7.21) differently from (7.12). The former handler (7.12d) of this exception did not account for possible inconsistency of the context caused by updating it with a presupposed proposition. Consequently, it did not account for conversational implicatures caused by this inconsistency. Moreover, in the new definition, the function upd is replaced with the more general abd.

7.2.4 Simplifying Dynamization Function

The discourse update function $\mathsf{dupd}_{\mathsf{g}}$ defined in 7.21 allows to simplify the definition of the dynamization function $\mathbb{D}$ defined in 4.27. Particularly, if $\mathsf{dupd}_{\mathsf{g}}$ is used, there is no longer need to update the context with the original static proposition when translating this static proposition into the dynamic equivalent. Definition (4.36b), repeated in (7.22), can be simplified to (7.23):

$$\mathbb{D}_o[\mathrm{P}] \doteq \lambda e\phi.\mathrm{P}e \wedge \phi(\mathsf{upd}(\mathrm{P}e, e)) \tag{7.22}$$

$$\mathbb{D}_o[\mathrm{P}] \doteq \lambda e\phi.\mathrm{P}e \wedge \phi e \tag{7.23}$$

This simplification is possible due to the addition of checkprovable function as the very first function called within gacc, which raises `UnprovablePropExc` when a stated proposition cannot be proven in the previous context, and to the capacity of the improved handler of `UnprovablePropExc` to abduce the proposition itself in the context. The conservation theorem proven in Section 4.3.3 also holds when (7.23) used instead of (7.22).

7.3 Presuppositions vs. Conversational Implicatures

Although presuppositions and conversational implicatures are different phenomena, it is sometimes hard to differentiate them. Analysis of Sections 7.1 and 7.2 based on the dynamic approach proposed in this dissertation helps to characterize the difference between presuppositions and implicatures more clearly.

As summarized in Table 7.1, presuppositions and implicatures are similar in some aspects and differ in others. The major difference is in their sources. Presuppositions are caused by particular lexical items, called presupposition triggers. Implicatures, however, occur by completely different means: they are side-effects of a context change.

	Presuppositions	**Implicatures$_1$**	**Implicatures$_2$**
Source	Lexical item (called trigger)	Assertion w.r.t. a context	Either assertion w.r.t. a context or presupposition accommodated in a context or another implicature accommodated in a context
Reason for exception raising	Failure in finding the proof of a proposition from the axioms contained in the context.		Unsatisfiable context.
Exception handling	Abducing what is missing in the proofs. If the abduced formula is exactly the proposition that was searched in the context and now is carried by the exception handler, it is a presupposition. All other abduced formulas are implicatures.		Removing those formulas from the context that are necessary for its refutation. Negating the removed formulas (these negated formulas are conversational implicatures). Adding these implicatures to the context.

Table 7.1: Comparison of presuppositions and two types of conversational implicatures.

Being part of the lexical meaning, presuppositions are always triggered. If they are already part of the context (or can be derived from it), the corresponding proposition is not added to the context.[5] In contrast, conversational implicatures are not pre-determined.

Table 7.1 distinguished two types of conversational implicatures. Implicatures of the first type are caused by assertions with respect to some context. They emerge as a result of abduction if the uttered assertion cannot be proved from

[5]This is often called "presupposition cancelling". However, as Geurts [1999] also observes, a presupposition can never be literally cancelled. It does always emerge and can sometimes be anaphorically linked to an antecedent in the context.

the axioms contained in the context. Therefore, this kind of implicature is similar to presupposition in that they are both related to proof failure. Perhaps this is the reason why they can be easily confused. Implicatures of the second type emerge only when the context becomes inconsistent and they solve this inconsistency.

Since presuppositions of a sentence are always triggered one can talk about presuppositions of a sentence independently of a context. However, one cannot talk about implicatures of an independent sentence. A sentence can have implicatures only with respect to some context.

The logical form of a sentence must be computed before it can be added to a discourse. Therefore, presuppositions, which are triggered by the sentence, must emerge before implicatures, which emerge only due to some changes in the context, i.e. during the discourse update. This answers Geurts' [1999] question:

> [...] why is it that conversational implicatures may cancel presuppositions and not the other way round? This question, which Gazdar doesn't answer, is especially urgent because one of the characteristic features of conversational implicatures is precisely that they are cancellable. [Geurts, 1999, p.66]

Propositions corresponding to presuppositions of a sentence go first into the context. Adding these propositions can lead to unsatisfiability of the context. This is when conversational implicatures (of the second type) come into force. One way to fix the unsatisfiability can be to remove a presupposed fact P from the context. The negation $\neg P$ of this fact is the conversational implicature of the sentence with respect to the context of the discourse being updated. This causes an impression of $\neg P$ cancelling P. In fact, the inconsistency caused by P leads to implicating that $\neg P$, therefore, a presupposition can cause implicatures. The other way around is simply impossible. A presupposition of a sentence cannot "cancel" its implicatures with respect to a context simply because by the time implicatures are computed, everything related to presuppositions of the sentence in the discourse has been already resolved once and for all. Geurts' intuition [1999] is very close to this explanation:

> [...] it would seem that presuppositions at least logically precede conversational implicatures, because they contribute to the proposition that an utterance expresses, on a given occasion, whereas conversational implicatures are derived from such propositions. In more authentically Gricean terms: while conversational implicatures are determined on the basis of 'what is said', presuppositions contribute to what is said by a speaker on a given occasion. [Geurts, 1999, p.66]

A common feature of presuppositions and conversational implicatures is that whether they emerge or not depends on the current context.[6] Their emergence can

[6]Note the difference between triggering and emergence of a presupposition. A presupposition

be implemented with exception raising and their accommodation in the context can be implemented with exception handling.

Consider an example from [Levinson, 1983, p. 187]. Sentence (62a) contains the presupposition triggering word *before*. When *before* is followed by a proposition, the proposition becomes presupposed. Therefore, Sentence (62a) presupposes (62b):

(62) a. *Sue cried before she finished her thesis.*

b. *She finished her thesis.*

However, if the verb *cried* is substituted with *died* in (62b), as shown in (63a), it seems that the presupposition is eliminated. In contrast, the sentence seem to convey the message (63b) that Sue did not finish the thesis.

(63) a. *Sue died before she finished her thesis.*

b. *She did not finish her thesis.*

Levinson gives the following informal explanation:

> [...] the statement of (86)[(63a)] asserts that the event of Sue's death precedes the (anticipated) event of her finishing her thesis; since we generally hold that people (and we assume Sue is a person) do not do things after they die, it follows that she could not have finished her thesis; this deduction from the entailments of the sentence together with background assumptions about mortals, clashes with the presupposition (85)[(62b)]; the presupposition is therefore abandoned in this context, or set of background beliefs [...] [Levinson, 1983, p. 187]

A more formal explanation of the difference between (62a) and (63a) is attempted below using framework GLχ. First of all, the interpretations of sentences should be computed from interpretations of their lexical items. All lexical interpretations now have the condition that a dynamic proposition does not update its context with its own assertion, as explained in Subsection 7.2.4. The lexical item *before* can cause a side effect of triggering two presuppositions. One presupposition is related to the content of the clause that syntactically precedes *before*. Another presupposition is related to the clause following *before*. Interpretation (7.24), where $\mathbf{P}, \mathbf{Q}$ are of type $\overline{o}$ and the term itself is of type $(\overline{o} \to \overline{o} \to \overline{o})$, takes both of these possible presuppositions into account:

$$\begin{aligned} &[\![\widetilde{before}]\!] \\ &= \lambda \mathbf{PQ}.\lambda e\phi.\mathbf{before}(\mathsf{checkvalid}(\mathbf{P}e(\lambda e.\top))e)(\mathsf{checkvalid}(\mathbf{Q}e(\lambda e.\top))e) \wedge \phi e \end{aligned} \tag{7.24}$$

is always triggered by its lexical item. Whether it emerges or not, however, depends on the context with respect to which it is evaluated. It emerges when it cannot be derived from this context.

Now, having all lexical interpretations, it is possible to compute the meanings of the sentences. The following example shows the major steps in the computation of the meaning of (62a).

EXAMPLE 7.9. [**S**, *Sue cried before she finished her thesis.*] The meaning of the sentence *Sue cried* is as follows:

$$\overline{[\![cried]\!]}\ \widetilde{[\![Sue]\!]} \rightarrow^*_{eval} \lambda e\phi.\mathbf{cried}(\mathsf{sel}(\text{named "Sue"})e) \wedge \phi e$$

The meaning of the sentence *She finished her thesis* is as follows:

$$\begin{aligned} &\overline{[\![finished]\!]}(\widetilde{[\![her]\!]}\overline{[\![thesis]\!]})\ \widetilde{[\![she]\!]} \\ \rightarrow^*_{eval}\ &\lambda e\phi.\mathbf{finished}(\mathsf{sel}(\lambda x.\mathbf{female}x)e) \\ &\qquad (\mathsf{sel}(\lambda x.\mathbf{thesis}x \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{female}y)e)x)e) \wedge \phi e \end{aligned}$$

Now it is possible to compute the meaning **S** of the sentence *Sue cried before she finished her thesis*:

$$\begin{aligned} \mathbf{S} = &\widetilde{[\![before]\!]}(\overline{[\![finished]\!]}\ \overline{[\![thesis]\!]}\ \widetilde{[\![she]\!]})(\overline{[\![cried]\!]}\ \widetilde{[\![Sue]\!]}) \\ \rightarrow^*_{eval}\ &\lambda e\phi.\mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}(\mathsf{sel}(\text{named "Sue"})e))e) \\ &\qquad (\mathsf{checkvalid}(\mathbf{finished}(\mathsf{sel}(\lambda x.\mathbf{female}x)e) \\ &\qquad\qquad (\mathsf{sel}(\lambda x.\mathbf{thesis}x \wedge \\ &\qquad\qquad\qquad \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{female}y)e)x)e))e) \wedge \phi e \end{aligned}$$

□

The next step is to update the meaning of the preceding discourse with the computed meaning of the new sentence, resolve presuppositional exceptions and accommodate presuppositions in the context (if necessary). Below this is done for the sentence (62a):

EXAMPLE 7.10. [$\mathsf{dupd}_{\mathsf{g}}$ **D S**] Assume that *Sue cried before she finished her thesis* is uttered in a discourse, which meaning **D** is defined in (7.25), with context c_{st}, shown in (7.26) and containing the following knowledge:

- there exists an individual named "Sue"
- there exists a thesis that belongs to Sue
- an individual cannot perform an action intrinstic to live individuals after dying
- to finish something is an action of live individuals

$$\mathbf{D} = \lambda\phi.\exists(\lambda t.(\text{named "Sue"}\, s) \wedge \exists(\lambda t.\mathbf{thesis}\ t \wedge \mathbf{poss}\ s\ t \wedge \phi \mathsf{c}_{st})) \tag{7.25}$$

$$\begin{aligned}\mathsf{c}_{st} = \mathsf{upd}&(\text{named "Sue"}\, s,\\ &\mathsf{upd}((\mathbf{thesis}\ t \wedge \mathbf{poss}\ s\ t),\\ &\quad \mathsf{upd}(\forall x \mathsf{a}.(C(\mathsf{a}) \rightarrow \neg(\mathsf{a}x \wedge \mathbf{before}(\mathbf{die}x, \mathsf{a}x))),\\ &\quad\quad \mathsf{upd}(C(\mathbf{finish}),\\ &\quad\quad\quad \top))))\end{aligned} \tag{7.26}$$

Updating $\mathbf{D}$ with $\mathbf{S}$ leads to the term (7.27):

$$\begin{aligned}&\mathsf{dupd}_{\mathsf{g}}\ \mathbf{D}\ \mathbf{S}\\ &= \lambda\phi.\ \mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi) && \text{(by (7.21a))}\\ &= \lambda\phi.\ (\lambda\phi.\exists(\lambda t.(\text{named "Sue"}\, s) \wedge \exists(\lambda t.\mathbf{thesis}\ t \wedge \mathbf{poss}\ s\ t \wedge \phi \mathsf{c}_{st})))\\ &\qquad (\lambda e.\mathsf{gacc}\ \mathbf{S}\ e\ \phi)\\ &\rightarrow^{*}_{eval} \lambda\phi.\ \exists(\lambda t.(\text{named "Sue"}\, s) \wedge \exists(\lambda t.\mathbf{thesis}\ t \wedge \mathbf{poss}\ s\ t \wedge \mathsf{gacc}\ \mathbf{S}\ \mathsf{c}_{st}\ \phi))\end{aligned} \tag{7.27}$$

The computation continues in the subterm $(\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}_{st}\ \phi)$ of (7.27). By (7.21b) gacc first attempts to compute $(\mathsf{checkprovable}\ \mathbf{S}\ \mathsf{c}_{st}\ \phi)$ which, by Definition (7.7), checks whether $\mathsf{c}_{st} \vdash \mathbf{S}\ \mathsf{c}_{st}\ (\lambda e.\top)$ holds. Term $(\mathbf{S}\ \mathsf{c}_{st}\ (\lambda e.\top))$ β-reduces as follows:

$$\begin{aligned}&\mathbf{S}\ \mathsf{c}_{st}\ (\lambda e.\top)\\ &= (\lambda e\phi.\mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}(\mathsf{sel}(\text{named "Sue"})e))e)\\ &\qquad (\mathsf{checkvalid}(\mathbf{finished}(\mathsf{sel}(\lambda x.\mathbf{female}x)e)\\ &\qquad\quad (\mathsf{sel}(\lambda x.\mathbf{thesis}x \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{female}y)e)x)e))e) \wedge \phi e)\ \mathsf{c}_{st}\ (\lambda e.\top)\\ &\rightarrow^{*}_{eval} \mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}(\mathsf{sel}(\text{named "Sue"})\mathsf{c}_{st}))\mathsf{c}_{st})\\ &\qquad (\mathsf{checkvalid}(\mathbf{finished}(\mathsf{sel}(\lambda x.\mathbf{female}x)\mathsf{c}_{st})\\ &\qquad\quad (\mathsf{sel}(\lambda x.\mathbf{thesis}x \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{female}y)\mathsf{c}_{st})x)\mathsf{c}_{st}))\mathsf{c}_{st})\end{aligned} \tag{7.28}$$

Since the context c_{st} contains knowledge about the existence of Sue and her thesis, the selection functions sel in (7.28) successfully returns corresponding variables leading to (7.29):

$$\mathsf{checkprovable}\ (\mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}s)\mathsf{c}_{st})(\mathsf{checkvalid}(\mathbf{finished}\ s\ t)\mathsf{c}_{st})) \tag{7.29}$$

However, checkvalid functions raise exceptions in (7.29). This is caused by the fact that the propositions $(\mathbf{cried}s)$ and $(\mathbf{finished}\ s\ t)$ cannot be proven based on the knowledge contained in the context c_{st}. Each of these exceptions is handled in turn according to (7.21d). Assume $(\texttt{UnprovablePropExc}\ (\mathbf{cried}s))$ is handled first:

$$\mathsf{gacc}\ \mathbf{S}\ (\mathsf{con}(\mathsf{abd}(\mathbf{cried}s, \mathsf{c}_{st}))\ \phi$$

$\mathsf{abd}(\mathbf{cried}s, \mathsf{c}_{st})$ abduces exactly $\mathbf{cried}s$ in c_{st} and the resulting context is satisfiable. Consequently, con returns this context, abbreviated below as c_{stc}:

$$\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}_{stc}\ \phi$$

The evaluation continues with computing $(\mathsf{checkprovable}\ \mathbf{S}\ \mathsf{c}_{stc}\ \phi)$, according to (7.21b), where it is checked whether $\mathsf{c}_{stc} \vdash \mathbf{S}\ \mathsf{c}_{stc}\ (\lambda e.\top)$ holds. However, the exception $(\texttt{UnprovablePropExc}\ (\mathbf{finished}\ s\ t))$ is raised as $(\mathbf{finished}\ s\ t)$ cannot be proven from the knowledge contained in the context c_{stc}. The exception is handled according to (7.21d):

$$\mathsf{gacc}\ \mathbf{S}\ (\mathsf{con}(\mathsf{abd}(\mathbf{finished}st, \mathsf{c}_{stc}))\ \phi$$

$\mathsf{abd}(\mathbf{finished}\ s\ t, \mathsf{c}_{stc})$ abduces $(\mathbf{finished}\ s\ t)$ in c_{stc} resulting in a satisfiable context. Consequently, con returns this satisfiable context, abbreviated below as c_{stcf}:

$$\mathsf{gacc}\ \mathbf{S}\ \mathsf{c}_{stcf}\ \phi$$

The evaluation proceeds with computing $(\mathsf{checkprovable}\ \mathbf{S}\ \mathsf{c}_{stcf}\ \phi)$, as defined in (7.21b). The $\mathsf{checkprovable}$ function checks whether $\mathsf{c}_{stcf} \vdash \mathbf{S}\ \mathsf{c}_{stcf}\ (\lambda e.\top)$ and, since there is no such a derivation, raises another exception $(\texttt{UnprovablePropExc}\ (\mathbf{before}(\mathbf{cried}s)(\mathbf{finished}\ s\ t)))$ (since now $\mathsf{checkvalid}$ is able to prove $(\mathbf{cried}s)$ and $(\mathbf{finished}\ s\ t)$ from the knowledge contained in the context c_{stcf}). The handler of this exception calls the abd function that abduces exactly $(\mathbf{before}(\mathbf{cried}s)(\mathbf{finished}\ s\ t))$ in c_{stcf}, leading to a new consistent context c_{stcfb}.
In the next call of gacc with arguments $\mathbf{S}$, c_{stcfb} and ϕ, $(\mathbf{S}\ \mathsf{c}_{stcfb}\ (\lambda e.\top))$ can be derived from c_{stcfb}. Therefore, $(\mathsf{checkprovable}\ \mathbf{S}\ \mathsf{c}_{stcfb}\ \phi)$ returns $(\mathbf{S}\ \mathsf{c}_{stcf}\ \phi)$ that β-reduces as follows:

$$\begin{aligned}
&\mathbf{S}\ \mathsf{c}_{stcfb}\ \phi\\
=\ &(\lambda e\phi.\mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}(\mathsf{sel}(\text{named "Sue"})e))e)\\
&\qquad(\mathsf{checkvalid}(\mathbf{finished}(\mathsf{sel}(\lambda x.\mathbf{female}x)e)\\
&\qquad\qquad(\mathsf{sel}(\lambda x.\mathbf{thesis}x \wedge \mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{female}y)e)x)e))e)\wedge\\
&\qquad\phi e)\mathsf{c}_{stcfb}\phi\\
\rightarrow^{*}_{eval}\ &\mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}(\mathsf{sel}(\text{named "Sue"})\mathsf{c}_{stcfb}))\mathsf{c}_{stcfb})\\
&\qquad(\mathsf{checkvalid}(\mathbf{finished}(\mathsf{sel}(\lambda x.\mathbf{female}x)\mathsf{c}_{stcfb})\\
&\qquad\qquad(\mathsf{sel}(\lambda x.\mathbf{thesis}x\wedge\\
&\qquad\qquad\quad\mathbf{poss}(\mathsf{sel}(\lambda y.\mathbf{female}y)\mathsf{c}_{stcfb})x)\mathsf{c}_{stcfb}))\mathsf{c}_{stcfb}) \wedge \phi\mathsf{c}_{stcfb}\\
=\ &\mathbf{before}(\mathsf{checkvalid}(\mathbf{cried}s)\mathsf{c}_{stcfb})(\mathsf{checkvalid}(\mathbf{finished}\ s\ t)\mathsf{c}_{stcfb}) \wedge \phi\mathsf{c}_{stcfb}
\end{aligned}$$

This time the two calls of the checkvalid function are able to prove (**cried**s) and (**finished** s t) from the knowledge contained in the context c_{stcfb}, returning the propositions themselves. This leads to (7.30):

$$\textbf{before}(\textbf{cried}s)(\textbf{finished}\ s\ t) \wedge \phi \mathsf{c}_{stcfb} \tag{7.30}$$

The final interpretation (7.31) of the the meaning of discourse **D** updated with the sentence *Sue cried before she finished her thesis* can be seen by combining terms (7.27) and (7.30):

$$\begin{aligned}
&\lambda\phi.\exists(\lambda t.(\text{named "Sue"}\ s) \wedge \\
&\quad \exists(\lambda t.\textbf{thesis}\ t \wedge \\
&\qquad \textbf{poss}\ s\ t \wedge \\
&\qquad\quad \textbf{before}\ (\textbf{cried}\ s)\ (\textbf{finished}\ s\ t) \wedge \\
&\qquad\quad \phi\ \mathsf{c}_{stcfb}))
\end{aligned} \tag{7.31}$$

The full form of the context c_{stcfb} in (7.30) is shown below:

$$\begin{aligned}
\mathsf{c}_{stcfb} = \mathsf{upd}(&\textbf{before}\ (\textbf{cried}\ s)\ (\textbf{finished}\ s\ t), \\
&\mathsf{upd}(\textbf{finished}\ s\ t, \\
&\ \mathsf{upd}(\textbf{cried}\ s, \\
&\ \ \mathsf{upd}(\text{named "Sue"}\ s, \\
&\ \ \ \mathsf{upd}(\textbf{thesis}\ t \wedge \textbf{poss}\ s\ t, \\
&\ \ \ \ \mathsf{upd}(\forall x\mathsf{a}.(C(\mathsf{a}) \rightarrow \neg(\mathsf{a}x \wedge \textbf{before}(\textbf{die}x, \mathsf{a}x))), \\
&\ \ \ \ \ \mathsf{upd}(C(\textbf{finish}), \top)))))))
\end{aligned} \tag{7.32}$$

Summing up, it has been computed that the sentence *Sue cried before she finished her thesis* presupposes that Sue cried and that Sue finished her thesis. The sentence does not implicate anything with respect to the context (7.26).

□

Now, recall Sentence (63a), *Sue died before she finished her thesis*. Analogously, to (62a), Sentence (63a) presupposes that Sue died and that Sue finished her thesis. The computation of these presuppositions is very similar to the computation of presuppositions for Sentence (62a) in Example (7.10) (the only difference is that **cried** is everywhere substituted for **died**). Example (7.11) shows interpretation $\mathbf{S}'$ of Sentence (63a) and major steps of updating **D** with $\mathbf{S}'$:

Example 7.11.

$$\begin{aligned}
\mathbf{S}' &= \widetilde{[\![before]\!]}(\overline{[\![finished]\!]}\ \overline{[\![thesis]\!]}\ \widetilde{[\![she]\!]})(\overline{[\![died]\!]}\ \widetilde{[\![Sue]\!]}) \\
&\rightarrow^*_{eval} \lambda e\phi.\textbf{before}(\mathsf{checkvalid}(\textbf{died}(\mathsf{sel}(\text{named "Sue"})e))e) \\
&\qquad (\mathsf{checkvalid}(\textbf{finished}(\mathsf{sel}(\lambda x.\textbf{female}x)e) \\
&\qquad\qquad (\mathsf{sel}(\lambda x.\textbf{thesis}x \wedge \\
&\qquad\qquad\quad \textbf{poss}(\mathsf{sel}(\lambda y.\textbf{female}y)e)x)e))e) \wedge \phi e
\end{aligned}$$

$$
\begin{aligned}
&\mathsf{dupd_g}\ \mathbf{D}\ \mathbf{S}' \\
&= \lambda\phi.\mathbf{D}(\lambda e.\mathsf{gacc}\ \mathbf{S}'\ e\ \phi) \\
&= \lambda\phi.(\lambda\phi.\exists(\lambda t.(\text{named "Sue"}\, s) \wedge \exists(\lambda t.\mathbf{thesis}\ t \wedge \mathbf{poss}\ s\ t \wedge \phi \mathsf{c}_{st})))\\
&\qquad (\lambda e.\mathsf{gacc}\ \mathbf{S}'\ e\ \phi) \\
&\rightarrow^*_{eval} \lambda\phi.\exists(\lambda s.(\text{named "Sue"}\, s) \wedge \exists(\lambda t.\mathbf{thesis}\ t \wedge \mathbf{poss}\ s\ t \wedge \mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}_{st}\ \phi))
\end{aligned}
\tag{7.33}
$$

Analogously to the previous example, where the same discourse is updated with Sentence (62a), there are several exceptions during the computation in the subterm ($\mathsf{gacc}\ \mathbf{S}'\ \mathsf{c}_{st}\ \phi$), particularly

- ($\mathsf{checkvalid}\ (\mathbf{died}\ s)\ \mathsf{c}_{st}$) raises ($\mathtt{UnprovablePropExc}\ (\mathbf{died}\ s)$). The handler abduces the fact ($\mathbf{died}\ s$) in c_{st}, leading to the consistent context c_{std}.
- ($\mathsf{checkvalid}\ (\mathbf{finished}\ s\ t)\ \mathsf{c}_{std}$) raises ($\mathtt{UnprovablePropExc}\ (\mathbf{finished}\ s\ t)$). The handler abduces the fact ($\mathbf{finished}\ s\ t$) leading to the consistent context c_{stdf}.
- ($\mathsf{checkprovable}\ (\mathbf{before}\ (\mathbf{died}\ s)\ (\mathbf{finsihed}\ s\ t))$) raises ($\mathtt{UnprovablePropExc}\ (\mathsf{checkprovable}(\mathbf{before}(\mathbf{died}\ s)(\mathbf{finsihed}\ s\ t))))$. The handler abduces the fact ($\mathbf{before}(\mathbf{died}\ s)(\mathbf{finsihed}\ s\ t)$) in e_{stdf} leading to the inconsistent, as shown below, context c_{stdfb}.

It is possible to refute the knowledge base of the context c_{stdfb} with the following proof:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{f}(s,t) \vdash \mathbf{f}(s,t)} \quad \dfrac{}{\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)) \vdash \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))}}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)) \vdash \mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))}\wedge_r \quad \dfrac{}{C(\mathbf{f}) \vdash C(\mathbf{f})}}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),\neg(\mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))) \vdash}\neg_l}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),C(\mathbf{f}) \rightarrow \neg(\mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))),C(\mathbf{f}) \vdash}\rightarrow_l}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),\forall x\mathtt{a}.(C(\mathtt{a}) \rightarrow \neg(\mathtt{a}(x) \wedge \mathbf{b}(\mathbf{d}(x),\mathtt{a}(x)))),C(\mathbf{f}) \vdash}\forall_l}{\mathbf{f}(s,t),\mathbf{d}(s),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),\mathsf{c}_{st} \vdash}w_l
$$

Therefore, exception ($\mathtt{InconsistentContextExc}\ \mathsf{c}_{stdfb}$) is raised. This exception is handled as defined in (7.21e). Thus, the makesat function chooses facts that cause unsatisfiability and removes them from the context. There are a few possibilities of which proposition can be removed to make the knowledge base in the context satisfiable. They are all highlighted in red in the proof below.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{f}(s,t) \vdash \mathbf{f}(s,t)} \quad \dfrac{}{\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)) \vdash \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))}}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)) \vdash \mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))}\wedge_r \quad \dfrac{}{C(\mathbf{f}) \vdash C(\mathbf{f})}}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),\neg(\mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))) \vdash}\neg_l}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),C(\mathbf{f}) \rightarrow \neg(\mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t))),C(\mathbf{f}) \vdash}\rightarrow_l}{\mathbf{f}(s,t),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),\forall x\mathtt{a}.(C(\mathtt{a}) \rightarrow \neg(\mathtt{a}(x) \wedge \mathbf{b}(\mathbf{d}(x),\mathtt{a}(x)))),C(\mathbf{f}) \vdash}\forall_l}{\mathbf{f}(s,t),\mathbf{d}(s),\mathbf{b}(\mathbf{d}(s),\mathbf{f}(s,t)),\mathsf{c}_{st} \vdash}w_l
$$

Two of them, particularly, the proposition $C(\mathbf{f})$ meaning that to finish something is an action of live objects and the formula $C(\mathbf{f}) \rightarrow \neg(\mathbf{f}(s,t) \wedge \mathbf{b}(\mathbf{d}(s), \mathbf{f}(s,t)))$ meaning that since the action of finishing something is the action of live objects, it is not possible for Sue to finish the thesis after dying. It is difficult to reject these facts. They are axioms.

Another possibility is to remove the proposition $\mathbf{b}(\mathbf{d}(s), \mathbf{f}(s,t))$. Since this proposition is actually what is asserted by the sentence, the removal of the proposition would mean that the hearer is (for some reason) not willing to accept the assertion itself.

The final possibility is the proposition $\mathbf{f}(s,t)$ (remember that it appeared in the context as a presupposition of the sentence). Removal of $\mathbf{f}(s,t)$ results in the implicature that Sue did not finish the thesis. This is exactly what is observed in (63b). Now, the new implicature can be added to the context, leading to the context c_{stdbn} shown in (7.34):[7]

$$\begin{aligned}\mathsf{c}_{stdbn} = \mathsf{upd}(&\neg\textbf{finished}\ s\ t,\\ &\mathsf{upd}(\textbf{before}\ (\textbf{died}\ s)\ (\textbf{finished}\ s\ t),\\ &\quad\mathsf{upd}(\textbf{died}\ s,\\ &\qquad\mathsf{upd}(\text{named “Sue”}\ s,\\ &\qquad\quad\mathsf{upd}(\textbf{thesis}\ t\ \wedge\ \textbf{poss}\ s\ t,\\ &\qquad\qquad\mathsf{upd}(\forall x\mathsf{a}.(C(\mathsf{a}) \rightarrow \neg(\mathsf{a}x \wedge \textbf{before}(\textbf{die}x, \mathsf{a}x))),\\ &\qquad\qquad\quad\mathsf{upd}(C(\textbf{finish}), \top)))))))\end{aligned} \tag{7.34}$$

Finally, evaluation of $\mathbf{S}'\ \mathsf{c}_{stdbn}\ \phi$ leads to (7.35):

$$\textbf{before}(\textbf{died}s)(\textbf{finished}st) \wedge \phi\mathsf{c}_{stdbn} \tag{7.35}$$

The final interpretation (7.36) of the the meaning of discourse $\mathbf{D}$ updated with the sentence *Sue <u>died</u> before she finished her thesis* can be seen by combining terms (7.33) and (7.35):

$$\begin{aligned}\lambda\phi.\exists(\lambda t.&(\text{named “Sue”}\ s) \wedge\\ &\exists(\lambda t.\textbf{thesis}\ t \wedge\\ &\quad\textbf{poss}\ s\ t \wedge\\ &\qquad\textbf{before}\ (\textbf{died}\ s)\ (\textbf{finished}\ s\ t) \wedge\\ &\qquad\phi\ \mathsf{c}_{stdbn}))\end{aligned} \tag{7.36}$$

Summing up, it has been computed that the sentence *Sue died before she finished her thesis* presupposes that Sue died and that Sue finished her thesis.

[7] It is also possible to make a simple negation of a fact leading to unsatisfiablity instead of removing it from the context and adding its negation. However, the former seem to be more natural.

However, since the latter presupposition causes the unsatisfiability of the context c_{stdfb}, it is erased and its negation is the conversational implicature of the sentence with respect to the context (7.26). □

Examples 7.10 and 7.11 illustrated an important distinction between the notions of presupposition and conversational implicatures. In these examples two syntactically similar sentences that differ only in one word were uttered in the same discourse but lead to different computation of the updated discourse. Initially, both sentences caused analogous side effects corresponding to presupposition triggering. These side effects were handled by abducing the presupposed facts into the context. The abduction of the presupposed facts corresponds to presupposition accommodation. The accommodation of the presupposed facts did not cause any additional side-effects in the case of the sentence *Sue cried before she finished her thesis* (Example 7.10). It did, however, in the case of the sentence *Sue died before she finished her thesis* (Example 7.11): the resulting context became inconsistent. Handling this inconsistency required removing a fact from the context; and the negation of this fact is the conversational implicature of this sentence with respect to the initial context.

As another example, consider the following sentence from [Beaver, 2001, p.144] and assume that it was uttered in an empty discourse:

(64) *If Bertha is not in the kitchen, then Anna realises that Bertha is in the attic.*

The factive verb *realises* triggers a presupposition that Bertha is in the attic: **in_attic** b. The presupposition is triggered in the "then"-clause of the conditional, therefore it should be evaluated with respect to the context made of the initial (empty) context and the assertion of the "if"-clause ($\neg$(**in_kitchen** b)). However,

$$\neg(\textbf{in_kitchen}\ b) \nvdash \textbf{in_attic}\ b$$

There are two possible ways to handle this. One of the ways is to abduce the presupposition itself (in blue):

$$\textbf{in_attic}\ b, \neg\ (\textbf{in_kitchen}\ b) \vdash \textbf{in_attic}\ b$$

This would mean that the presupposition of the sentence is accommodated in the given empty context.

Another possibility is to abduce the following fact (in blue):

$$\neg\ (\textbf{in_kitchen}\ b) \rightarrow \textbf{in_attic}\ b, \neg\ (\textbf{in_kitchen}\ b) \vdash \textbf{in_attic}\ b$$

The abduced fact is the conversational implicature that can be spelled as "if Bertha is not in the kitchen, then Bertha is in the attic".

Which of these two possibilities should be preferred by the system is a matter of pragmatic constraints.

This chapter informally presented some possible extensions of framework GLχ (gradually developed in Chapters 4, 5 and 6) for handling presuppositions (particularly, presuppositions triggered by factive verbs) and conversational implicatures. On the one hand, the chapter illustrated that the framework has a potential to be adapted for even more complex natural language phenomena and presented general informal analyses of some of these phenomena in GLχ. On the other hand, the chapter assumed that solutions to many issues needed for a complete formalization of the discussed phenomena are provided. Among them are the formalization of the abduction function abd and the formalization of the makesat function that chooses which facts to be removed from unsatisfiable context to make it satisfiable. Moreover, a suitable theorem prover should stand behind functions checkvalid and checkprovable. These issues are possible directions for future research.

Chapter 8

Conclusions

The goal of showing that semantics of discourse dynamics can be compositionally expressed with standard well-established mathematical methods was successfully achieved. The proposed framework GLχ expresses dynamic meanings using only simply-typed lambda calculus and classical logic and they are obtained with a precisely defined translation from equivalent static lexical interpretations.

8.1 Summary of Results

Taking de Groote's [2006] continuation-based frameworks G_0 (Section 4.1) and G (Section 4.2) as the basis, this dissertation gradually extended them to a substantially more expressive continuation-based framework with exceptions.

Section 4.3 defined the framework GL. Context in this framework has a more realistic structure than in G_0 and G and can contain knowledge (viewed, for example, as a conjunction of formulas). Moreover, contexts are treated in GL in a more sophisticated way (e.g. dynamic individuals are defined as functions from contexts) and information can be extracted from contexts based on given descriptive contents. Moreover, a conservation result that guarantees that everything expressible in static semantics can be expressed in GL is proved.

Chapter 5 illustrated on examples how dynamic meanings of lexical items are constructed and how dynamic meanings of sentences are computed. While conventional lexical items can be transformed into their dynamic counterparts simply by applying precisely defined dynamization rules, interpretations of unconventional items require additional modifications expressing their potential to cause linguistic side-effects. This potential is formalized with an exception raising mechanism and Section 5.2 showed how this mechanism is employed for interpreting familiarity presuppositions triggered by referring expressions. Regardless of whether dynamic terms interpret conventional items or items with potential side-effects, they (and their compositions) can be represented compactly in a form

reminiscent of original Montague's interpretation.

A potential side-effect of a sentence comes into force, i.e. an exception is raised, when its interpretation is evaluated with respect to some context. A discourse has a concrete context in GL and thus, when an interpretation of a discourse is updated with an interpretation of a sentence, this context is passed as one of the arguments to the interpretation of the sentence (due to the employment the continuation-passing technique). Then, if the sentence has, for example, a referring expression with a descriptive content P (formalized as a property), its interpretation can raise an exception depending whether the provided context contains an individual that matches the property P or not. If there is such an individual in the context, it is simply retrieved and is used in the remaining computation. Otherwise, an exception carrying property P is raised, propagated and handled by the discourse update function. The handler of this exception introduces an existentially quantified variable that satisfies P and updates the context with this new knowledge. This corresponds to global accommodation of a familiarity presupposition.

Chapter 6 defined the framework GLχ, which extends the language and calculus of GL with constructors and evaluation rules related to exception raising and handling. Framework GLχ successfully handles the presupposition projection problem in conditionals and the binding problem.

Chapter 7 envisaged how the framework can be extended even further and illustrated the possibility to handle semantic and (some) pragmatic phenomena in a unified framework. It described how presuppositions triggered by the factive verb *know* and presuppositions triggered by *before* can be interpreted; and outlined an approach to deal with some conversational implicatures. Finally, it demonstrated that it is possible to draw an intuitive comparison between presuppositions and conversational implicatures using the terminology of GLχ.

8.2 Comparison with Related Work

Below GLχ is compared with related work on formal semantics, specifically with Discourse Representation Theory, Dynamic Predicate Logic and Dynamic Montague Grammar, reviewed in Chapter 2. For the comparison, the simpler definitions of GLχ's restricted versions G and GL, defined in Chapter 4, suffice.

8.2.1 Discourse Representation Theory

Since framework GL defines meanings of natural language expressions in a language that is a superset of the language of first-order logic, the translation $\ominus$ of discourse representation structures into first-order logic formulas, shown in Subsection 2.2.1 and repeated below, can be used in GL in a straightforward way.

Translation $\ominus$ from DRT to FOL:

- $\{\{x_1, \ldots, x_n\}\{C_1, \ldots, C_m\}\}^{\ominus} \doteq \exists x_1, \ldots x_n.C_1^{\ominus} \wedge \cdots \wedge C_m^{\ominus}$
- $C^{\ominus} \doteq C$
- $(\neg C)^{\ominus} \doteq \neg C^{\ominus}$

For example, the DRS (8.1) can be easily incorporated in a expression in GL in the following way:

$$\begin{aligned} &\lambda e\phi.\{\{x_1, \ldots, x_n\}\{C_1, \ldots, C_m\}\}^{\ominus} \wedge \phi e^* \\ \doteq\ &\lambda e\phi.\exists x_1, \ldots x_n.C_1^{\ominus} \wedge \cdots \wedge C_m^{\ominus} \wedge \phi e^* \end{aligned}$$

where e^* is the environment e possibly updated with knowledge related to variables $x_1, \ldots, x_n$. Note that since ϕ is applied to e^*, the content contained in the DRS falls in the scope of continuation of the λ-term.

$$\begin{array}{|c|}\hline x_1 \ldots x_n \\ \hline C_1 \\ \vdots \\ C_m \\ \hline \end{array} \tag{8.1}$$

Moreover, framework GL is capable of simulating DRT's accessibility constraints. Recall, for example, the meaning (5.66), repeated below in (8.2) (assuming that the anaphora is resolved, i.e. that sel returns y), of Sentence (65):

(65) *Every farmer who owns a donkey beats it.*

$$\lambda e\phi.\forall(\lambda x.\mathbf{f}x \rightarrow \forall(\lambda y.(\mathbf{d}y \wedge \mathbf{o}xy) \rightarrow \mathbf{b}xy)) \wedge \phi e \tag{8.2}$$

Note that the context that is given to the continuation ϕ as an argument is just a variable e of type γ. It is not updated with variables x and y and properties related to them. Since the context provided to the continuation does not contain x and y, they are not accessible for the selection functions sel possibly occurring in the interpretations of new sentences of the discourse. This directly corresponds to the accessibility constraint in DRT.

8.2.2 Dynamic Predicate Logic

As mentioned in Section 2.2, DPL, at least as it is presented in [Groenendijk and Stokhof, 1991] does not provide a translation from natural language to the language of DPL. In contrast, this dissertation is focused exactly on providing a translation from natural language to the language of continuation based-dynamic logic. This makes it impossible to make a direct comparison between DPL and

GLχ (as well as its predecessors G and GL). Nevertheless, it is still possible to compare the frameworks by defining certain translations between their different levels. Since, among G, GL and GLχ, the closest to DPL w.r.t. expressive power is G, translations between G and DPL are defined subsequently.

Consider Figure 8.1. The missing translation h_{DPL} of natural language into the language of DPL is represented by the dashed arrow. The translation h_G of natural language into the language G_L of continuation-based dynamic logic is represented by the green arrow.

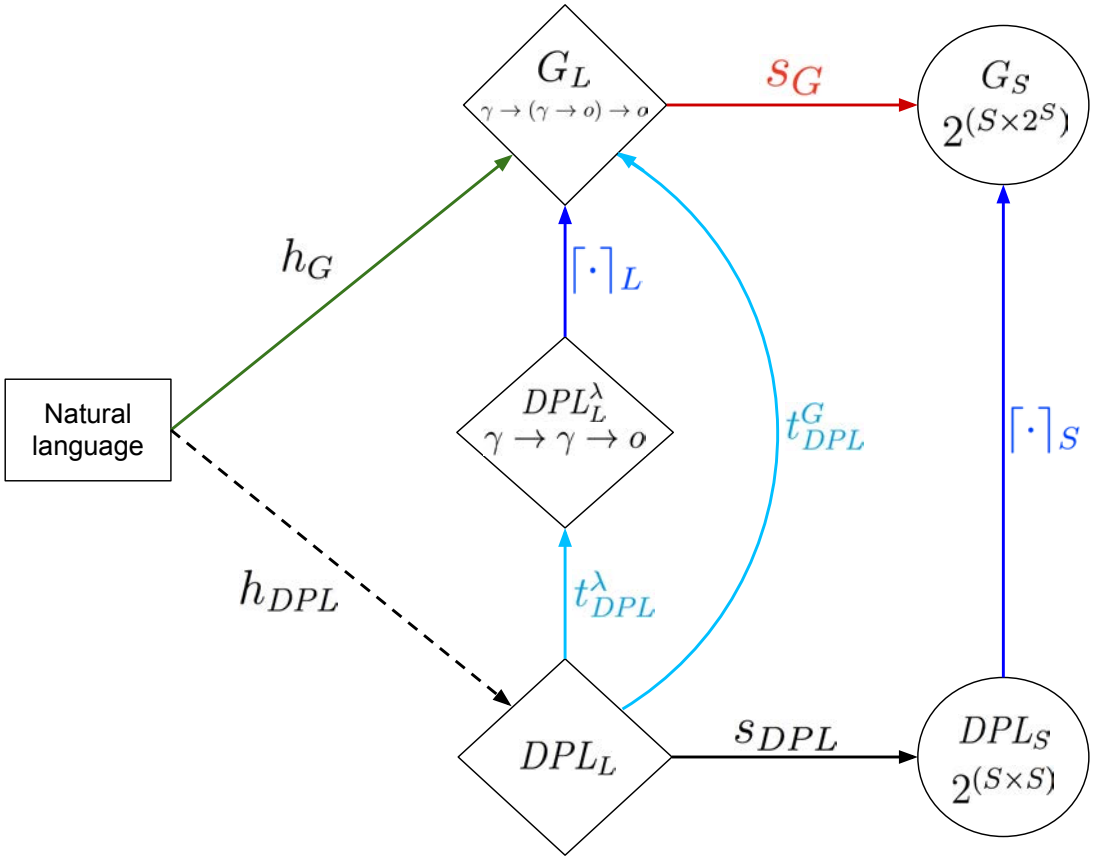


Figure 8.1: Comparison between DPL and G.

What Groenendijk and Stokhof [1991] define is a translation s_{DPL} of the language of DPL into binary relations on states DPL_S. This translation is illustrated with the black solid arrow.

There are two possible ways to compare DPL and G. One possibility is to define a translation s_G (red arrow) of the language of G into relations on states G_S and to compare G_S with DPL_S. Another possibility is to define a translation t^{λ}_{DPL} (shorter light blue arrow) of the language DPL_L of DPL into a new language DPL^{λ}_L and to compare DPL^{λ}_L with the language G_L of G. These two comparisons are presented in more details below.

Comparison 1

DPL interprets propositions as binary relations on states and its semantics is defined as the following translation from DPL_L to DPL_S (solid black arrow):

DEFINITION 8.1. [Translation s_{DPL}] Let A, B and P be terms in DPL_L of type o. Let h, g and k be assignments of values to variables (a.k.a. states). Translation $s_{DPL} : DPL_L \to DPL_S$ is defined as follows:

$$\begin{aligned}
(A)_{s_{DPL}} &= \{\langle g, h\rangle \mid h = g \wedge h \models A\} \\
(\neg P)_{s_{DPL}} &= \{\langle g, h\rangle \mid h = g \wedge \neg(\exists k.\ \langle h, k\rangle \in (P)_{s_{DPL}})\} \\
(A \wedge B)_{s_{DPL}} &= \{\langle g, h\rangle \mid \exists k.\ \langle g, k\rangle \in (A)_{s_{DPL}} \wedge \langle k, h\rangle \in (B)_{s_{DPL}}\} \\
(\exists i.P)_{s_{DPL}} &= \{\langle g, h\rangle \mid \exists k.\ k[i]g \wedge \langle k, h\rangle \in (P)_{s_{DPL}}\}
\end{aligned}$$

Thus, for every proposition P, $(P)_{s_{DPL}}$ is in $2^{(S\times S)}$.
A similar translation s_G (red arrow) of propositions in G_L into relations defined on states can be defined:

DEFINITION 8.2. [Translation s_G] Let S, the set of states, be the semantic domain that interprets γ. The translation $s_G : G_L \to G_S$ is defined as follows:

$$\begin{aligned}
(A)_{s_G} &= \{\langle g, H\rangle \mid g \in H \wedge g \models A\} \\
(\sim P)_{s_G} &= \{\langle g, H\rangle \mid g \in H \wedge \langle g, S\rangle \notin (P)_{s_G}\} \\
(A \curlywedge B)_{s_G} &= \{\langle g, H\rangle \mid \langle g, \{h \mid \langle h, H\rangle \in (B)_{s_G}\}\rangle \in (A)_{s_G}\} \\
(\Sigma_i x.P)_{s_G} &= \{\langle g, H\rangle \mid \exists x.\langle (x, i) :: g, H\rangle \in (P)_{s_G}\}
\end{aligned}$$

Note that s_G interprets propositions not just as relations between states, but as relations betweens states and sets of states: for each proposition P, $(P)_{s_G}$ is in $2^{(S\times 2^S)}$. Therefore, s_G is richer than s_{DPL} and avoids existentially quantified states and equality relations on states by using set inclusion. Moreover, there exists a canonical embedding (larger blue arrow) of DPL_S into G_S:

DEFINITION 8.3. [Embedding of DPL_S into G_S] Let R be set of pair of states (in $2^{(S\times S)}$) in DPL_S, the embedding $\lceil\cdot\rceil_S$ of DPL_S into G_S is as follows:

$$\lceil R\rceil_S \doteq \{\langle g, H\rangle \mid \exists h.\ h \in H \wedge \langle g, h\rangle \in R\}$$

$\lceil R\rceil_S$ is in $2^{(S\times 2^S)}$.

Note that there is a correspondence between types constructed from γ and o and sets constructed from S and the cross product powerset operations:

$$\begin{aligned}
[\gamma] &\approx S \\
[\alpha_1 \to \alpha_2 \to \cdots \to \alpha_n \to o] &\approx 2^{[\alpha_1]\times[\alpha_2]\times\cdots\times[\alpha_n]}
\end{aligned}$$

Consequently, the type $(\gamma \to (\gamma \to o) \to o)$ corresponds to the set $2^{(S\times 2^S)}$.

The following translation t^G_{DPL} (longer light blue arrow) of DPL_L to G_L can be defined:

DEFINITION 8.4. [Translation t^G_{DPL}] Let A, B and P be propositions. The translation $t^G_{DPL} : DPL_L \to G_L$ is defined as follows:

$$\begin{aligned}(A)_{t^G_{DPL}} &= A\\ (\neg P)_{t^G_{DPL}} &= \sim (P)_{t^G_{DPL}}\\ (A \wedge B)_{t^G_{DPL}} &= (A)_{t^G_{DPL}} \curlywedge (B)_{t^G_{DPL}}\\ (\exists i.P)_{t^G_{DPL}} &= \Sigma_i x.(P)_{t^G_{DPL}}\end{aligned}$$

THEOREM 8.5. *Let P be a proposition. Then the following equivalence holds:*

$$\lceil (P)_{s_{DPL}} \rceil_S = ((P)_{t^G_{DPL}})_{s_G}$$

by postulating

$$g[i]h \quad \textit{iff} \quad \exists x.(h = (x, i) :: g)$$

Proof. The proof is by induction on the structure of P. □

By Theorem 8.5, transforming DPL_L to G_L and interpreting the result in G_S according to s_G is equivalent to DPL's original semantics.

Comparison 2

Each proposition A in DPL_L can be translated into a term of type $(\gamma \to \gamma \to o)$ in DPL^λ_L in a way that reflects the semantics of DPL. The translation is given by the following definition:

DEFINITION 8.6. [Translation t^λ_{DPL}] Let T of type o be an atomic proposition, P, A, and B of type o be propositions, g, h and k be variables of type γ. The translation $t^\lambda_{DPL} : DPL_L \to DPL^\lambda_L$ is defined as follows:

$$(T)_{t^\lambda_{DPL}} \doteq \lambda gh.\ h = g \wedge T \tag{8.3a}$$

$$(\neg P)_{t^\lambda_{DPL}} \doteq \lambda gh.\ h = g \wedge \neg(\exists k.(P)_{t^\lambda_{DPL}} hk) \tag{8.3b}$$

$$(A \wedge B)_{t^\lambda_{DPL}} \doteq \lambda gh.\ \exists k.\ (A)_{t^\lambda_{DPL}} gk \wedge (B)_{t^\lambda_{DPL}} kh \tag{8.3c}$$

$$(\exists x.P)_{t^\lambda_{DPL}} \doteq \lambda gh.\ \exists k.\ k[x]g \wedge (P)_{t^\lambda_{DPL}} kh \tag{8.3d}$$

The resulting language DPL^λ_L can be compared with G_L.

Note that DPL^λ_L has the existential quantifier in the definitions of negation (8.3b) and conjunction (8.3c) and a subterm with equality in the definitions of the atomic proposition (8.3a) and negation (8.3b). DPL uses these existential quantifications and equalities in order to constraint the pairs of states that represent the final interpretation. Because of that, the final interpretation in DPL^λ_L would have chains of equalities of the form $\cdots \wedge g = g' \wedge \cdots \wedge g' = g'' \wedge \ldots$ and

some deductive reasoner would be required for establishing the final interpretation of a sentence. In G, however, there is no need for this kind of equalities and existential quantification. All necessary restrictions that natural language poses on a model are interpreted using β-reduction only.

A canonical embedding (smaller blue arrow) of DPL^{λ}_{L} into G_L can be defined:

Definition 8.7. [Embedding of DPL^{λ}_{L} into G_L] Let A be a term of type $(\gamma \to \gamma \to o)$ in DPL^{λ}_{L}, the embedding $\lceil\cdot\rceil_L$ of DPL^{λ}_{L} into G_L is as follows:

$$\lceil A\rceil_L \doteq \lambda e\phi.\exists e'.\ \phi\ e' \wedge A\ e\ e'$$

$\lceil A\rceil_L$ is of type $(\gamma \to (\gamma \to o) \to o)$.

Theorem 8.8. *Let A, B and P be terms of type o in DPL_L. Then the following logical equivalences hold:*

$$\begin{aligned}
\lceil (A)_{t^{\lambda}_{DPL}}\rceil_L &\equiv \overline{A}\\
\lceil (\neg P)_{t^{\lambda}_{DPL}}\rceil_L &\equiv\ \sim \lceil (P)_{t^{\lambda}_{DPL}}\rceil_L\\
\lceil (A \wedge B)_{t^{\lambda}_{DPL}}\rceil_L &\equiv \lceil (A)_{t^{\lambda}_{DPL}}\rceil_L \curlywedge \lceil (B)_{t^{\lambda}_{DPL}}\rceil_L
\end{aligned}$$

Proof. The proof is by induction on the structure of terms. □

An analogous logical equivalence does not hold, however, for the existential quantifier: $\lceil (\exists x.P)_{t^{\lambda}_{DPL}}\rceil_L$ reduces to (8.4) and $\Sigma\lceil (P)_{t^{\lambda}_{DPL}}\rceil_L$ reduces to (8.5):

$$\lceil (\exists x.P)_{t^{\lambda}_{DPL}}\rceil_L = \lambda e\phi.\ \exists e'.\ \phi\ e' \wedge \exists k.\ k[x]e \wedge \lceil (P)_{t^{\lambda}_{DPL}}\rceil_L\ k\ e' \tag{8.4}$$

$$\Sigma\lceil (P)_{t^{\lambda}_{DPL}}\rceil_L = \lambda e\phi.\ \exists e'.\ \phi\ e' \wedge \exists x.\ \lceil (P)_{t^{\lambda}_{DPL}}\rceil_L\ (x :: e)\ e' \tag{8.5}$$

Nevertheless, terms (8.4) and (8.5) are similar from the point of view of constraining their interpretation in a model. Thus, the constraint in (8.4) that k is interpreted as e except for the values of x is expressed in (8.5) simply by explicit update of the context e with x. Note, moreover, that x is a free variable in (8.4), whereas (8.5) is a closed term. Therefore, interpretation (8.5) is advantageous over (8.4).

8.2.3 Dynamic Montague Grammar

Among dynamic approaches to discourse semantics, DMG is the closest to G, as it uses λ-terms to express meaning and computes the meaning of complex expressions by application of the λ-terms representing the meanings of the lexical items of this expression. Moreover, propositions in DMG are defined as terms of type $((s \to t) \to t)$, i.e. they have an argument that functions as a continuation, a term from states to propositions. Indeed, functioning as parameters for the

model-theoretical interpretation of formulas, states of DMG are reminiscent of the context in G. However, the important difference is that DMG implements this dependency on a parameter on the model-theoretical level. In framework G, in contrast, this is done in a more syntactic way, i.e. during compositional translation of natural language into logical language. Thus, the phenomena of natural language are reflected in GL in the resulting logical formula independently of a model in which it is to be interpreted.

The similarity and difference of DMG and G can be seen in the basic definitions of the frameworks. Below some of these definitions are compared in detail.

Dynamic Conjunction

Recall the dynamic conjunction ; (2.47) in DMG, repeated below in (8.6). This dynamic conjunction amounts to the composition of the conjuncts. Compare it with the dynamic conjunction (4.25c) in G, unfolded in (8.7) using the equation (4.24b). It is also defined as a composition of the conjuncts. However, this composition is done within the standard lambda calculus.

$$\mathbf{A};\mathbf{B} \doteq \lambda p.\mathbf{A}(^{\wedge}(\mathbf{B}p)) \tag{8.6}$$

$$\overline{\mathrm{A}} \curlywedge \overline{\mathrm{B}} \doteq \lambda e\phi.\mathbf{A}e(\lambda e'.\mathbf{B}e'\phi) \tag{8.7}$$

Dynamization of a Proposition

Recall Equation (2.45), repeated below in (8.8), defining how a proposition is dynamized in DMG. Compare (8.8) with the definition (4.25a) of dynamization of a proposition A in G, repeated below in (8.9):

$$\uparrow\mathrm{A} \doteq \lambda p.\mathrm{A} \wedge^{\vee} p \tag{8.8}$$

$$\overline{\mathrm{A}} \doteq \lambda e\phi.\mathrm{A} \wedge \phi e \tag{8.9}$$

Note that, while in (8.9) the context appears explicitly as the variable e, in (8.8) the state is implicit. Moreover, applying the $^{\vee}$-operator to the variable p corresponds to applying the continuation ϕ to the (possibly updated) context e. Therefore, while in DMG the continuation p is applied to the current state implicitly via type lowering, in G it is done by standard functional application. Note also that in G the context is computed, along with the whole logical formula, compositionally during the translation of a natural language expression to the logical formula.

Cross-Sentential and Donkey Anaphora

Recall equivalence (2.50) in DMG, repeated in (8.10), which is an important feature for dealing with cross-sentential and donkey anaphora:

$$(\mathcal{E}d.\uparrow\mathrm{A});\uparrow\mathrm{B} =_{\beta} \mathcal{E}d.(\uparrow\mathrm{A};\uparrow\mathrm{B}) \tag{8.10}$$

Lexical item	Dynamic interpretation in DMG	Dynamic interpretation in GL
farmer	$\lambda x.\uparrow\mathbf{f}x$	$\overline{\mathbf{f}}$
donkey	$\lambda x.\uparrow\mathbf{d}x$	$\overline{\mathbf{d}}$
owns	$\lambda\mathbf{Y}y.(^{\vee}\mathbf{Y}(^{\wedge}\lambda x.\uparrow\mathbf{o}xy))$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{o}}\mathbf{xy}))$
beats	$\lambda\mathbf{Y}y.(^{\vee}\mathbf{Y}(^{\wedge}\lambda x.\uparrow\mathbf{b}xy))$	$\lambda\mathbf{YX}.\mathbf{X}(\lambda\mathbf{x}.\mathbf{Y}(\lambda\mathbf{y}.\overline{\mathbf{b}}\mathbf{xy}))$
a	$\lambda\mathbf{PQ}.\mathcal{E}d_2.(^{\vee}\mathbf{P}d_2;^{\vee}\mathbf{Q}d_2)$	$\lambda\mathbf{PQ}.\overline{\exists}(\lambda\mathbf{x}.\mathbf{Px}\mathbin{\overline{\wedge}}\mathbf{Qx})$
every	$\lambda\mathbf{PQ}.\mathcal{A}d_1.(^{\vee}\mathbf{P}d_1\Rightarrow^{\vee}\mathbf{Q}d_1)$	$\lambda\mathbf{PQ}.\overline{\forall}(\lambda\mathbf{x}.\mathbf{Px}\mathbin{\overline{\Rightarrow}}\mathbf{Qx})$
who	$\lambda\mathbf{RQ}x.(^{\vee}\mathbf{Q}x;^{\vee}\mathbf{R}(^{\wedge}(\lambda\mathbf{P}.^{\vee}\mathbf{P}x)))$	$\lambda\mathbf{RQx}.\mathbf{Qx}\mathbin{\overline{\wedge}}\mathbf{R}(\lambda\mathbf{P}.\mathbf{Px})$
it	$\lambda\mathbf{Q}.^{\vee}\mathbf{Q}d_2$	$\lambda\mathbf{P}.\mathbf{P}(\mathsf{sel}(\lambda x.\mathbf{non_human}x))$

Table 8.1: Comparison of dynamic interpretations of lexical items of the sentence *Every farmer who owns a donkey beats it* in DMG and GL.

As Proposition 4.16 proves and examples in Chapter 5 illustrate, an analogous property, particularly Equation (8.11), holds in G:

$$\Sigma(\lambda x.\overline{\mathrm{A}[x]})\between\overline{\mathrm{B}}=_{\beta}\Sigma(\lambda x.\overline{\mathrm{A}[x]\between\overline{\mathrm{B}}}) \tag{8.11}$$

Compare the normal forms (2.48) and (4.28), repeated below in (8.12a) and (8.12b), of (8.10) and (8.11) respectively:

$$\lambda p.\exists x.\{x/d\}(\mathrm{A}\wedge(\mathrm{B}\wedge{}^{\vee}p)) \tag{8.12a}$$
$$\lambda e\phi.\exists(\lambda x.\mathrm{A}[x]\wedge(\mathrm{B}\wedge\phi(x::e))) \tag{8.12b}$$

Note, that the existentially quantified variable x in (8.12b) is already in $FV(\mathrm{A}[x])$. In contrast, the state switcher $\{x/d\}$ in (8.12a) still needs to be pushed inside the logical formula to substitute the unbound discourse marker by the bound variable x. Moreover, note that since terms in DMG have unbound variables (discourse markers), additional care should be taken when two formulas are composed in order to avoid destructive assignment.

(Lexical) Interpretations

Consider, once again, the donkey sentence (66):

(66) *Every farmer who owns a donkey beats it.*

Table 8.1 compares the DMG's lexical interpretations of the donkey sentence and the compact lexical interpretations in GL (introduced in Table 5.4).

Note that lexical interpretations in GL are more intuitive. Moreover, they are independent of indexing, unlike terms in DMG. Indeed, the use of implicit states in DMG complicates, if not excludes, the integration of anaphora resolution in a compositional way. Thus, DMG, as DRT and DPL, assumes that anaphora is resolved before the actual computation of meaning. In contrast, GL is designed in a

way that anaphora can be resolved compositionally. The framework is capable of integrating an anaphora resolution mechanism and is designed to compositionally provide the necessary content (i.e. descriptive content and context) to this mechanism to resolve the anaphora. Therefore, there is no need in GL for any kind of special symbols (such as discourse markers) and there is no need for indexing them or any other variables in advance.

Terms in GL can be directly functionally applied to each other in accordance with syntactic parsing. This is advantageous, since it makes compositionality much simpler in GL.

8.3 Advantages of the Developed Framework

- The approach is truly compositional: the meaning of a complex expression is computed by β-reducing the composition, obtained by functional application, of the meanings of its lexical items.

- Continuation-based dynamic logic serves as an "intermediate" logical language for an "indirect" model-theoretical interpretation of sentences of a natural language. Therefore, there is no need to be concerned with model-theoretical issues when translating a natural language sentence into a logical sentence, since the resulting logical formula can be interpreted with classical well-studied model theory, if desired.

- The approach is independent of the "intermediate" language used to express meanings of the expressions and, hence, is general. This allows to use mathematical and logical theories developed outside computational linguistics. Therefore, natural language phenomena can be explained in terms of well-established and well-understood theories.

- The logical language can be used for reasoning problems that are difficult to solve directly on the natural language.

- Variables do not have any special status and are variables in the usual mathematical sense. Therefore, the notions of free and bound variables are standard.

- There are no imperative dynamic notions, such as assignment functions. Therefore, the destructive assignment problem does not occur. Meanings assigned to expressions are closed λ-terms.

- There is no need for rules that artificially extend the scope of quantifiers.

- Context and content are regarded separately, but they do interact during the computation of the meaning of discourse.

- The approach treats semantic and pragmatic phenomena in a unified way.
- The approach does not depend on any specific structure given to the context. In contrast, context is defined as a term of type parameter γ and, therefore, its structure can be altered when necessary.[1]
- The context is represented by a term, and, hence, it is explicit. Therefore, the troubles related to implicit states are avoided.
- The framework allows the integration of any desired anaphora resolution algorithm (into the selection function sel). Since it is a part of the interpretations of anaphoric lexical items and it receives the current context as one of its arguments, anaphora is always resolved compositionally during the whole computation.
- Triggers of various linguistic side-effects can be expressed by modifying the standard dynamic interpretations of lexical items of the same syntactic category in a way that potentially raises exceptions. Then the appropriate exception handlers can be added to take these additional linguistic side-effects into consideration.

Taken together, the advantages above lead to a very important feature of the framework, namely flexibility. It can be extended for handling more complex phenomena of natural language and, therefore, opens opportunities for future development.

8.4 Future Work

The potential of the framework for being extended to handle many other complex linguistic phenomena is, perhaps, as valuable as the applications of the framework that have been described here so far. Chapter 7 outlines the possible extensions for handling presuppositions triggered by the factive verb *know* and presuppositions triggered by *before*. A possible direction for further research is to interpret other presupposition triggers in a similar manner. Moreover, Chapter 7 sketches how to handle two types of conversational implicatures, and the further development of this sketch is another exciting direction for further research. These two broad directions are among the possibilities opened up by this dissertation.

[1] In [de Groote, 2006], focused on cross-sentential and donkey anaphora, the context is defined as a list of individuals (i.e. $\gamma \doteq \texttt{list of}\ \iota$). In [de Groote and Lebedeva, 2010], where a compositional framework for handling presuppositions triggered by definite descriptions is proposed, it is defined as a list of pairs "individual $\times$ proposition" (i.e. $\gamma \doteq \texttt{list of}\ (\iota \times o)$). This dissertation exploited an environment consisting of propositions (i.e. $\gamma \doteq \texttt{list of}\ o$). More complex natural language phenomena would require more sophisticated definitions of γ and, therefore, representations of context.

Appendix A

Lambda Calculus

In this Appendix, based on [Barendregt, 1981], [Barendregt, 1992] and [Hindley and Seldin, 2008], the basic definitions and theorems of type-free (A.1) and simply-typed (A.2) lambda calculus are presented.

A.1 Type-free Lambda Calculus

Definition A.1. [λ-terms] The set of **λ-terms** Λ constructed from an enumerable set of variables $V = \{v, v_1, v_2, \dots\}$ is defined inductively as follows:

$$\begin{array}{rcll} x \in V & \Longrightarrow & x \in \Lambda & \\ M, N \in \Lambda & \Longrightarrow & (MN) \in \Lambda & (\textbf{application}) \\ x \in V, M \in \Lambda & \Longrightarrow & (\lambda x.M) \in \Lambda & (\textbf{abstraction}) \end{array}$$

In the term $(\lambda x.M)$, called abstraction, the variable x is the **argument** of the function and M is the **body** of the function.

Example A.2. The following are λ-terms:

$$\begin{array}{l} x \\ (x_1 x_2) \\ (\lambda x.(x_1 x_2)) \\ (\lambda x_1.(x_1 x_2)) \\ ((\lambda x.(x_1 x_2))x_3) \end{array}$$

□

Remark A.3. [Parenthesis conventions]

- application is left-associative

$$MN_1N_2 \dots N_n =_s (\dots((MN_1)N_2)\dots N_n)$$

- a sequence of λ-abstractions $\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.M)))$ is abbreviated as $\lambda x_1 x_2 \ldots x_n.M$

$$\lambda x_1 x_2 \ldots x_n.M =_s \lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.M)))$$

- parentheses surrounding the body of an abstraction can be dropped

$$\lambda x_1 x_2 \ldots x_n.(M) =_s \lambda x_1 x_2 \ldots x_n.M$$

- outermost parentheses can be dropped

$$(M) =_s M$$

Note that according to the conventions on parentheses, term $\lambda x.MN$ is a more concise way of writing $\lambda x.(MN)$ and is not equivalent to $(\lambda x.M)N$.

Example A.4. According to Remark A.3, the λ-terms in Example A.2 can be written as follows:

$$x$$
$$x_1 x_2$$
$$\lambda x.x_1 x_2$$
$$\lambda x_1.x_1 x_2$$
$$(\lambda x.x_1 x_2)x_3$$

□

Definition A.5. [Free and bound variables] A variable x is **free** in a λ-term M if x is not in the scope of λx. If x is in the scope of λx, it is **bound**.

Example A.6. In the term $(\lambda x.x_1 x_2)$, the variables x_1 and x_2 are free. In the term $(\lambda x_1.x_1 x_2)$, the variable x_1 is bound and the variable x_2 is free. In the term $x(\lambda x.x)$, the variable occurs free in the subterm x and bound in the subterm $\lambda x.x$. □

Definition A.7. [Closed λ-terms]

1. The set of **free variables** of M, $FV(M)$ is defined inductively as follows:

$$\begin{aligned} FV(x) &= \{x\} \\ FV(\lambda x.M) &= FV(M) - \{x\} \\ FV(MN) &= FV(M) \cup FV(N) \end{aligned}$$

2. M is **closed** or a **combinator**, if $FV(M) = \emptyset$

If an equation $M = N$ is provable in the lambda calculus, the provability is denoted by $\lambda \vdash M = N$ or sometimes just by $M = N$.

Definition A.8. [Axioms and rules] For all $M, N, L, Z \in \Lambda$ the following axioms and rules hold:

$$\frac{}{(\lambda x.M)N = M[x := N]}\ \beta\text{-conversion}$$

$$\frac{}{M = M}$$

$$\frac{M = N}{N = M}$$

$$\frac{M = N \qquad N = L}{M = L}$$

$$\frac{M = N}{MZ = NZ}$$

$$\frac{M = N}{ZM = ZN}$$

$$\frac{M = N}{\lambda x.M = \lambda x.N}\ \text{rule } \xi$$

Importantly, substitution $[x := N]$ in M, denoted $M[x := N]$, is only applicable to the free occurrences of x in M. For example,

$$(xy(\lambda x.x))[x := N] = Ny(\lambda x.x)$$

Definition A.9. [Substitution] The result of **substitution** of N for the free occurences of x in M, i.e. $M[x := N]$, is defined inductively on the structure of M as follows:

$$\begin{aligned}
x[x := N] &\doteq N \\
y[x := N] &\doteq y \ \text{ provided } x \neq_s y \\
(\lambda y.M_1)[x := N] &\doteq \lambda y.(M_1[x := N]) \\
(M_1M_2)[x := N] &\doteq (M_1[x := N])(M_2[x := N]) \\
(\lambda x.M_1)[x := N] &\doteq \lambda x.M_1
\end{aligned}$$

Lemma A.10. [Substitution lemma] *If $x \neq_s y$ and $x \notin FV(L)$, then*

$$M[x := N][y := L] =_s M[y := L][x := N[y := L]]$$

Proof. The proof is by induction on the structure of M. □

When performing a substitution $M[x := N]$, it is necessary to rename those bound variables in M that are free in N. Otherwise, the substitution may lead to a false result. For example, without renaming the bound variable x in $\lambda x.xy$, the substitution $(\lambda x.xy)[y := x]$ leads to the term $\lambda x.xx$ that acts differently from the desired term, because the free variable x became bound. However, changing x to z, for example, before making the substitution, leads to the desired term $\lambda z.zx$.

Definition A.11. [α-conversion] A **change of bound variable** x in M, or an **α-conversion** in M, is the replacement of an occurrence of $\lambda x.N$ in M by $\lambda y.(N[x := y])$, where y does not occur in N.

Definition A.12. [α-congruency] M and N are **α-congruent**, denoted $M =_\alpha N$, if one can result from the other by a finite series of changes of bound variables.

Example A.13.

$$\lambda x.xy =_\alpha \lambda z.zy \neq_\alpha \lambda x.xx$$
$$\lambda xy.yx(\lambda x.x) =_\alpha \lambda xy.yx(\lambda z.z) =_\alpha \lambda zy.yz(\lambda x.x)$$
$$\lambda xy.yx =_\alpha \lambda zy.yz =_\alpha \lambda zx.xz =_\alpha \lambda yx.xy$$

□

It is natural to identify the terms that are α-congruent, as they represent the same processes. Moreover, the same processes can be represented by different terms. For example, $\lambda x.Mx$ and M both lead to MN when applied to N. Hence, the following rule can be introduced:

Definition A.14. [Extensionality] **Extensionality** is the following derivation rule, provided $x \notin FV(MN)$:

$$\frac{Mx = Nx}{M = N}$$

The extensionality rule allows to prove $\lambda x.Mx = M$. Alternatively, $\lambda x.Mx = M$ can be considered to be an axiom:

Definition A.15. [η-conversion] Let $x \notin FV(M)$. Then

$$\frac{}{\lambda x.Mx = M}\ \eta\text{-conversion}$$

Definition A.16. [β-normal form]

1. M is a **β-normal form**, if M has no subterm of the form $(\lambda x.L)K$

2. M **has a β-normal form**, if there exists an N such that N is a β-normal form and $N = M$.

When M is a β-normal form, it is often said that M is in normal form.

Example A.17.

1. $\lambda x.x$ is in normal form.
2. $(\lambda x.x)y$ has a normal form, namely y.
3. $(\lambda xy.y)z$ has a normal form, namely $(\lambda y.y)$.
4. $(\lambda xy.x)z$ has a normal form, namely z.
5. $(\lambda x.xx)(\lambda y.yy)$ has no normal form.

□

A notion of reduction on Λ is a binary relation on Λ. The classical notion of reduction β is defined as follows:

Definition A.18. $\beta = \{((\lambda x.M)N, M[x := N]) | M, N \in \Lambda\}$

Definition A.19. [β-redex, β-contractum] A β**-redex** is a term M such that $(M, N) \in \beta$ for some term N. In this case N is called β**-contractum** of M.

Definition A.20. The notion of reduction β induces the following binary relations:

$\rightarrow_\beta$ **one-step β-reduction**
$\rightarrow_\beta^*$ **β-reduction**
$=_\beta$ **β-equality** (also called **β-convertibility**)

These relations are inductively defined as follows:

$$\begin{array}{rcl}
(M, N) \in \beta & \Longrightarrow & M \rightarrow_\beta N \\
M \rightarrow_\beta N & \Longrightarrow & ZM \rightarrow_\beta ZN \\
M \rightarrow_\beta N & \Longrightarrow & MZ \rightarrow_\beta NZ \\
M \rightarrow_\beta N & \Longrightarrow & \lambda x.M \rightarrow_\beta \lambda x.N \\
 & & \\
M \rightarrow_\beta N & \Longrightarrow & M \rightarrow_\beta^* N \\
M \rightarrow_\beta^* M & & \\
M \rightarrow_\beta^* N, N \rightarrow_\beta^* Z & \Longrightarrow & M \rightarrow_\beta^* Z \\
 & & \\
M \rightarrow_\beta^* N & \Longrightarrow & M =_\beta N \\
M =_\beta N & \Longrightarrow & N =_\beta M \\
M =_\beta N, N =_\beta Z & \Longrightarrow & M =_\beta Z
\end{array}$$

The notions of β-redex and β-equality allow to give an alternative, more formal, definition of β-normal form:

DEFINITION A.21. [β-normal form]

1. A term N is called a β-normal form, if N does not contain (as subterm) any β-redex.
2. A term N is a β-normal form of M, if N is a β-normal form and $M =_\beta N$.

The notion of β-reduction is very important, because it characterizes provability in λ and it is Church-Rosser:

PROPOSITION A.22. *$M =_\beta N$ iff $\lambda \vdash M = N$*

Proof. See [Barendregt, 1981, p.59]. □

THEOREM A.23. [Church-Rosser theorem for $\rightarrow_\beta^*$] *If $L \rightarrow_\beta^* M$ and $L \rightarrow_\beta^* N$, then there exists a term Z such that $M \rightarrow_\beta^* Z$ and $N \rightarrow_\beta^* Z$.*

Proof. See [Barendregt, 1981, p.62] or [Hindley and Seldin, 2008, p.289]. □

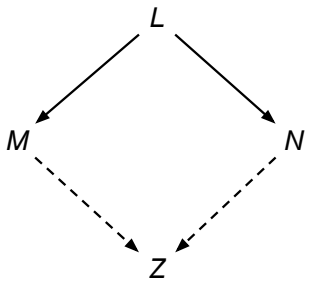


Figure A.1: Diamond property.

The Church-Rosser theorem says that for two β-convertible terms, there is a term to which they both β-reduce, as illustrated in Figure A.1. The property described in the theorem, that if a term can be reduced to two different terms, then these two terms can be further reduced to one term, is called the **diamond property** or **confluence**. The theorem states that β-reduction is confluent.

A.2 Simply-typed Lambda Calculus

Lambda terms can be assigned expressions, called "types", to denote their intended input and output sets. There exists two typing paradigms: à la Curry, sometimes called **implicit**, and à la Church, sometimes called **explicit**. This section first recalls the basics of Curry-style approach and then briefly compares it with the Church-style.

DEFINITION A.24. [Simple types] Given a set A of **atomic types**, the set of **types** T is inductively defined as follows:

$$\begin{array}{rcl} \alpha \in A & \Longrightarrow & \alpha \in T \\ \alpha, \beta \in A & \Longrightarrow & (\alpha \to \beta) \in T \quad (\textbf{function types}) \end{array}$$

An atomic type is intended to denote some particular set. A function type $(\alpha \to \beta)$ is intended to denote some set of functions from α to β, i.e. the functions that take as the argument a member of the set denoted by α and return as an output a member of the set denoted by β.

REMARK A.25. [Parenthesis convention] A complex functional type $(\alpha_1 \to (\alpha_2 \to \cdots \to (\alpha_{n-1} \to \alpha_n) \ldots))$ is abbreviated as $\alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$ (i.e. parentheses are associated to the right):

$$(\alpha_1 \to (\alpha_2 \to \cdots \to (\alpha_{n-1} \to \alpha_n) \ldots)) =_s \alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$$

DEFINITION A.26. [$\lambda\to$-Curry]

1. A **statement** is of the form $M : \sigma$ with $M \in \Lambda$ and $\sigma \in T$. The type σ is the **predicate** and the term M is the **subject** of the statement.
2. A **declaration** is a statement with a variable as a subject.
3. A **basis** is a set of declarations with distinct variables as subjects.

DEFINITION A.27. [Derivation rules in $\lambda\to$-Curry] A statement $M : \sigma$ is **derivable** from a basis Γ, denoted $\Gamma \vdash_{\lambda\to\text{-Curry}} M : \sigma$, $\Gamma \vdash_{\lambda\to} M : \sigma$ or simply $\Gamma \vdash M : \sigma$, if $\Gamma \vdash M : \sigma$ can be produced by the following rules:

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{axiom}$$

$$\frac{\Gamma \vdash M : \alpha \to \beta \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}\ \text{app}$$

$$\frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta}\ \text{abs}$$

LEMMA A.28. [Substitution lemma for $\lambda\to$-Curry]

1. *If* $\Gamma \vdash M : \sigma$, *then* $\Gamma[\alpha := \tau] \vdash M : \sigma[\alpha := \tau]$.
2. *Suppose* $\Gamma, x : \sigma \vdash M : \tau$ *and* $\Gamma \vdash N : \sigma$. *Then* $\Gamma \vdash M[x := N] : \tau$.

Proof.

1. The proof is by induction on the derivation of $M : \sigma$.

2. The proof is by induction on the generation of $\Gamma, x : \sigma \vdash M : \tau$

□

The following theorem states that the set of terms having a certain type is closed under reduction:

THEOREM A.29. [Subject reduction theorem for $\lambda\!\rightarrow$-Curry] *Suppose* $M \rightarrow_\beta^* N$. *Then*

$$\Gamma \vdash M : \sigma \implies \Gamma \vdash N : \sigma$$

Proof. See [Barendregt, 1992, p.41]. □

While in Curry's approach each term is assigned a type after the term has been built, in Church's approach, the type of a term is integrated in the term itself. For example, the term $\lambda x.x$ can be assigned a type according to the Curry and Church styles respectively as follows:

$$\vdash_{Curry} \lambda x.x : (\sigma \rightarrow \sigma)$$
$$\vdash_{Church} \lambda x^\sigma.x : (\sigma \rightarrow \sigma)$$

The term $\lambda x^\sigma.x$ itself is annotated in a Church system by σ. This means that $\lambda x^\sigma.x$ takes the argument x from the particular set denoted by σ. In contrast, a Curry system allows each term to have a polymorphic type. For example, the term $\lambda x.x : (\sigma \rightarrow \sigma)$ denotes the operation of doing nothing regardless how σ is instantiated: it can stand, for example, for integers or for booleans.

DEFINITION A.30. [T-annotated λ-terms] Let V be a set of variables, T be a set of types. The set Λ_T of **T-annotated λ-terms** is defined as follows:

$$\begin{aligned} x \in V &\implies x \in \Lambda_T \\ M, N \in \Lambda_T &\implies MN \in \Lambda_T \\ x \in V, M \in \Lambda_T, \sigma \in T &\implies \lambda x^\sigma.M \in \Lambda_T \end{aligned}$$

The typed lambda calculus à la Church is defined similarly to the typed lambda calculus à la Curry: an important difference is in the derivation rule corresponding to the abstraction: the abstracted variable is explicitly annotated with a type in the Church-style system. The explicit annotation of types in Church-style system makes it possible to decide whether a term has a certain type. This is an undecidable question for some Curry systems. On the other hand, a Curry-style system has more power and more flexibility than a Church-style system. For example, the easiest way to answer the question whether an untyped term M has any typed analogues is to re-state the question in Curry's notation. Furthermore, Curry-style systems can be generalized in ways Church-style systems cannot.

Terms à la Church can be easily mapped into terms à la Curry. This is done simply by "erasing" all type annotations within the term à la Church:

DEFINITION A.31. $|\cdot| : \Lambda_T \to \Lambda$ is defined as follows:

$$\begin{array}{rcl} |x| & \doteq & x \\ |MN| & \doteq & |M||N| \\ |\lambda x^\sigma.M| & \doteq & \lambda x.|M| \end{array}$$

The following proposition states that terms in the Church version project to terms in the Curry version of $\lambda\!\to$; and that terms in the Curry style can be "lifted" to terms in the Church style:

PROPOSITION A.32.

1. *Let* $M \in \Lambda_T$. *Then*

$$\Gamma \vdash_{Church} M : \sigma \implies \Gamma \vdash_{Curry} |M| : \sigma$$

2. *Let* $N \in \Lambda$. *Then*

$$\begin{array}{rcl} \Gamma \vdash_{Curry} N : \sigma & \implies & \textit{exists } M \in \Lambda_T \\ & & \textit{such that } \Gamma \vdash_{Church} M : \sigma \textit{ and } |M| =_\alpha N \end{array}$$

Proof. Both (1) and (2) are proved by induction on the given derivation. □

See [Barendregt, 1992] and [Hindley and Seldin, 2008] for profound introductions to the two typing styles and their detailed comparisons.

Bibliography

[Ajdukiewicz, 1935] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935. [cited in page: 25]

[Aristotle, 350 BCE] Aristotle. *Prior Analytics*. 350 B.C.E. [cited in page: 240]

[Asher and Pogodalla, 2010a] Nicholas Asher and Sylvain Pogodalla. A montagovian treatment of modal subordination. In *SALT 20*, Vancouver, Canada, 2010. [cited in page: 29, 121]

[Asher and Pogodalla, 2010b] Nicholas Asher and Sylvain Pogodalla. SDRT and continuation semantics. In *Logic and Engineering of Natural Language Semantics (LENLS)*, volume 7, 2010. [cited in page: 121]

[Asher, 1993] Nicholas Asher. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers, 1993. [cited in page: 2, 70]

[Barendregt, 1981] Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, 1981. [cited in page: 271, 276]

[Barendregt, 1992] Henk Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II. Oxford University Press, 1992. [cited in page: 271, 278, 279]

[Barker and Shan, 2008] Chris Barker and Chung-chieh Shan. Donkey anaphora is in-scope binding. *Semantics and Pragmatics*, 1(1):1–46, June 2008. [cited in page: 90]

[Barker, 2002] Chris Barker. Continuations and the nature of quantification. *Natural Language Semantics*, 10(3):211–242, 2002. [cited in page: 29, 90]

[Barker, 2004] Chris Barker. Continuations in natural language. In Hayo Thielecke, editor, *Proceedings of the Fourth ACM SIGPLAN Continuations Workshop (CW'04)*, Birmingham, UK, 2004. [cited in page: 90]

[Beaver, 2001] David Beaver. *Presupposition and Assertion in Dynamic Semantics*. CSLI Publications, Stanford, CA, 2001. [cited in page: 54, 154, 257]

[Bernardi and Moortgat, 2010] Raffaella Bernardi and Michael Moortgat. Continuation semantics for the Lambek-Grishin calculus. *Information and Computation*, 208(5):397–416, 2010. [cited in page: 29]

[Brasoveanu, 2007] Adrian Brasoveanu. *Structured Nominal and Modal Reference*. PhD thesis, The State University of New Jersey, 2007. [cited in page: 70]

[Burton-Roberts, 1989] Noel Burton-Roberts. *The Limits to Debate: A Revised Theory of Semantic Presupposition*. Cambridge University Press, 1989. [cited in page: 54]

[Chang and Keisler, 1990] C. C. Chang and H. Jerome Keisler. *Model theory*. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, third edition, 1990. [cited in page: 11]

[Chierchia, 1992] Gennaro Chierchia. Anaphora and dynamic binding. *Linguistics and Philosophy*, 15(2):111–183, 1992. [cited in page: 2]

[Church, 1940] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, (5):56–68, 1940. [cited in page: 93]

[Curtis *et al.*, 2005] Jon Curtis, Gavin Matthews, and David Baxter. On the effective use of Cyc in a question answering system. In *IJCAI Workshop on Knowledge and Reasoning for Answering Questions*, 2005. [cited in page: 120]

[Davidson, 1967] Donald Davidson. Truth and meaning. *Synthese*, 17:304–323, 1967. [cited in page: 6]

[de Groote and Lebedeva, 2010] Philippe de Groote and Ekaterina Lebedeva. Presupposition accommodation as exception handling. In *Proceedings of the SIGDIAL 2010 Conference*, Tokyo, Japan, September 2010. Association for Computational Linguistics. [cited in page: 121, 269]

[de Groote and Pogodalla, 2004] Philippe de Groote and Sylvain Pogodalla. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information*, 13(4):421–438, 2004. [cited in page: 30]

[de Groote, 2001a] Philippe de Groote. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155, 2001. [cited in page: 29]

[de Groote, 2001b] Philippe de Groote. Type raising, continuations, and classical logic. In R. van Rooy and M. Stokhof, editors, *Thirteenth Amsterdam Colloquium*, pages 97–101. Institute for Logic, Language and Computation, Universiteit van Amsterdam, December 2001. [cited in page: 90]

[de Groote, 2002] Philippe de Groote. Tree-adjoining grammars as abstract categorial grammars. In *Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 145–150. Università di Venezia, 2002. [cited in page: 30]

[de Groote, 2006] Philippe de Groote. Towards a montagovian account of dynamics. In *Semantics and Linguistic Theory XVI*, 2006. [cited in page: 2, 29, 91, 94, 96, 100, 105, 106, 109, 112, 116, 119, 121, 142, 143, 164, 165, 181, 259, 269]

[Dekker, 1999] Paul Dekker. Scopes in discourse. *Language and Computation*, 1:7–32, 1999. [cited in page: 2]

[Dever, 2006] Josh Dever. *Compositionality*. Clarendon Press, 2006. [cited in page: 8]

[Dowty *et al.*, 1981] David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics*. D. Reidel, Dordrecht, 1981. [cited in page: 23, 27]

[Egli and von Heusinger, 1995] Urs Egli and Klaus von Heusinger. The epsilon operator and E-type pronouns. In Urs Egli, Peter E. Pause, Christoph Schwarze, Arnim von Stechow, and Götz Wienold, editors, *Lexical Knowledge in the Organization of Language (Current Issues in Linguistic Theory 114)*, pages 121–141. Benjamins, Amsterdam, 1995. [cited in page: 48]

[Elbourne, 2005] Paul Elbourne. *Situations and Individuals*. MIT Press, 2005. [cited in page: 154]

[Fitting,] Melvin Fitting. Intensional logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2011 edition. [cited in page: 44]

[Frege, 1892] Gottlob Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und Philosophische Kritik 100*, pages 25–50, 1892. [cited in page: 11, 15, 21]

[Frege, 1914] Gottlob Frege. Letter to Jourdain. In G. Gabriel et al, editor, *Philosophical and Mathematical Correspondence*, pages 78–80. Chicago University Press, 1980, 1914? [cited in page: 10]

[Frege, 1952] Gottlob Frege. On sense and reference. In Peter Geach and Max Black, editors, *Translations from the Philosophical Writings of Gottlob Frege*, pages 56–78. Basil Blackwell, Oxford, 1952. [cited in page: 11, 16, 21, 56]

[Frege, 1963] Gottlob Frege. Compound thoughts. *Mind*, 72(285):1–17, 1963. [cited in page: 10]

[Gamut, 1991] L. T. F. Gamut. *Logic, language, and meaning*, volume I. University of Chicago Press, 1991. [cited in page: 42]

[Gazdar, 1979] Gerald Gazdar. *Pragmatics: Implicature, Presupposition and Logical Form*. Academic Press, New York, 1979. [cited in page: 54, 61]

[Geach, 1962] Peter Thomas Geach. *Reference and Generality*. Cornell University Press, Ithaca, New York, 1962. [cited in page: 49]

[Geurts, 1997] Bart Geurts. Good news about the description theory of names. *Journal of Semantics*, 14(4):319–348, 1997. [cited in page: 154]

[Geurts, 1999] Bart Geurts. *Presuppositions and Pronouns*, volume 3 of *CRiSPI*. Elsevier, Amsterdam, 1999. [cited in page: 60, 248, 249]

[Girard, 1987] Jean-Yves Girard. Linear logic. *Theorerical Computer Science*, 50:1–102, 1987. [cited in page: 30]

[Grätzer, 1979] George Grätzer. *Universal Algebra*. Springer, New York, 2nd edition, 1979. [cited in page: 8]

[Grice, 1975] Herbert Paul Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics*, volume 3, pages 41–58. Elsevier, 1975. [cited in page: 61, 240]

[Groenendijk and Stokhof, 1990] Jeroen Groenendijk and Martin Stokhof. Dynamic Montague grammar. In L. Kalman and L. Polos, editors, *Proceedings of the Second Symposion on Logic and Language*, pages 3–48. Eotvos Lorand University Press, 1990. [cited in page: 2, 42, 45, 70, 71, 72, 76, 77, 78, 79]

[Groenendijk and Stokhof, 1991] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100, 1991. [cited in page: 2, 42, 45, 70, 71, 261, 262]

[Harel *et al.*, 1984] David Harel, Dexter Kozen, and Jerzy Tiuryn. Dynamic logic. In *Handbook of Philosophical Logic*, volume II, pages 497–604. MIT Press, 1984. [cited in page: 42]

[Heim, 1982] Irene Heim. *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, University of Massachusetts at Amherst, 1982. [cited in page: 2, 42, 45, 48, 49, 62, 91]

[Heim, 1983] Irene Heim. File Change Semantics and the Familiarity Theory of Definites. In Rainer Bäuerle, Christoph Schwarze, and Arnim von Stechow, editors, *Meaning, Use and Interpretation of Language*, pages 164–189. Walter de Gruyter, Berlin, 1983. [cited in page: 2, 45]

[Hindley and Seldin, 2008] J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators, an Introduction*. Cambridge University Press, 2008. [cited in page: 271, 276, 279]

[Hintikka and Sandu, 1997] Jaakko Hintikka and Gabriel Sandu. Game-theoretic semantics. In Johan van Benthem and Alice ter Meulen, editors, *The Handbook of Logic and Language*, chapter 6, pages 361–410. Elsevier, Amsterdam, 1997. [cited in page: 42]

[Hintikka, 1962] Jaakko Hintikka. *Knowledge and Belief. An Introduction to the Logic of the Two Notions*. Cornell University Press, 1962. [cited in page: 44]

[Hobbs *et al.*, 1993] Jerry R. Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. Interpretation as abduction. *Artificial Intelligence*, 63(1–2), 1993. [cited in page: 241]

[Hobbs, 2004] Jerry R. Hobbs. Abduction in natural language understanding. In Laurence R. Horn and Gregory Ward, editors, *The Handbook of Pragmatics*. Blackwell, 2004. [cited in page: 240, 241]

[Hodges, 1997] Wilfrid Hodges. *A shorter model theory*. Cambridge University Press, Cambridge, 1997. [cited in page: 11]

[Jacobson, 1997] Pauline Jacobson. The syntax/semantics interface in categorial grammar. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, chapter 4, pages 89–116. Blackwell, 1997. [cited in page: 24]

[Janssen, 1997] Theo M. V. Janssen. Compositionality. In Johan van Benthem and Alice ter Meulen, editors, *The Handbook of Logic and Language*, chapter 7, pages 417–473. Elsevier, Amsterdam, 1997. [cited in page: 10]

[Kamp and Reyle, 1993] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Studies in Linguistics and Philosophy. Springer, July 1993. [cited in page: 2, 45, 63, 69]

[Kamp, 1981] Hans Kamp. A theory of truth and semantic representation. In Jeroen Groenendijk, Theo Janssen, and Martin Stokhof, editors, *Formal Methods in the Study of Language, Part 1*, volume 135, pages 277–322. Mathematical Centre Tracts, Amsterdam, 1981. Reprinted in Jeroen Groenendijk, Theo Janssen and Martin Stokhof (eds), 1984, Truth, Interpretation, and Information; Selected Papers from the Third Amsterdam Colloquium, Foris, Dordrecht, pp. 1–41. [cited in page: 2, 42, 45, 62, 63, 69, 91]

[Kaplan, 1970] David Kaplan. On the logic of demostratives. *Journal of Philosophical Logic*, 8:81–98, 1970. [cited in page: 44, 45]

[Karttunen, 1973] Lauri Karttunen. Presuppositions of Compound Sentences. *Linguistic Inquiry*, 4:167–193, 1973. [cited in page: 54, 61]

[Karttunen, 1974] Lauri Karttunen. Presuppositions and Linguistic Context. *Theoretical Linguistics*, 1:181–194, 1974. [cited in page:]

[Kripke, 1963] Saul A. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. [cited in page: 44]

[Lambek, 1958] Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958. [cited in page: 26]

[Lambek, 1961] Joachim Lambek. On the calculus of syntactic types. In R. Jacobson, editor, *Structure of Language and its Mathematical Aspects, Proceedings of the Symposia in Applied Mathematics*, volume XII, pages 166–178. American Mathematical Society, 1961. [cited in page: 26]

[Lebedeva and Woltzenlogel Paleo, 2010] Ekaterina Lebedeva and Bruno Woltzenlogel Paleo. Using proofs to compute implicatures. In *CiE 2010 Conference. Abstract and Handout Booklet*, Ponta Delgada, Azores, Portugal, June/July 2010. [cited in page: 240]

[Lenat, 1995] D. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11), 1995. [cited in page: 120]

[Levinson, 1983] Stephen Levinson. *Pragmatics*. Cambridge University Press, Cambridge, 1983. [cited in page: 55, 56, 250]

[Lewis, 1969] David Lewis. *Convention*. Harvard University Press, 1969. [cited in page: 120]

[Lewis, 1970] David Lewis. General semantics. *Synthese*, 22(1-2):18–67, 1970. [cited in page: 27, 44]

[Lewis, 1979] David Lewis. Scorekeeping in a language game. *Journal of Philosophical Logic*, 8(1):339–359, 1979. [cited in page: 42, 55]

[Lillibridge, 1999] Mark Lillibridge. Unchecked exceptions can be strictly more powerful than call/cc. *Higher-Order and Symbolic Computation*, (5):275–307, 1999. [cited in page: 203]

[Martin and Pollard, 2010] Scott Martin and Carl Pollard. Hyperintensional dynamic semantics: Analyzing definiteness with enriched contexts. *Lecture Notes in Computer Science*, 2010. to appear. [cited in page: 96, 105, 121]

[Martin and Pollard, 2011] Scott Martin and Carl Pollard. A higher-order theory of presupposition. *Special Issue on Logic and Natural Language*, 2011. to appear. [cited in page: 105, 121]

[Milner *et al.*, 1997] Robin Milner, Mads Tofte, and David Macqueen. *The Definition of Standard ML*. MIT Press, Cambridge, MA, USA, 1997. [cited in page: 148]

[Montague, 1968] Richard Montague. Pragmatics. In R. Klibansky, editor, *Contemporary Philosophy*, volume 1, pages 102–122. La Nuova Italia Editrice, 1968. [cited in page: 43, 95]

[Montague, 1970a] Richard Montague. English as a formal language. In B. Visentini et. al., editor, *Linguaggi nella e nella Tecnica*, pages 189–224. Edizioni di Comunità, Milan, 1970. [cited in page: 1, 22, 23, 41, 143]

[Montague, 1970b] Richard Montague. Universal grammar. In *Theoria*, pages 373–398. 1970. [cited in page: 1, 22, 23, 41, 143]

[Montague, 1973] Richard Montague. The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language: proceedings of the 1970 Stanford workshop on Grammar and Semantics*, pages 221–242. Reidel, Dordrecht, 1973. [cited in page: 1, 22, 23, 24, 26, 27, 35, 41, 90, 143]

[Moortgat, 1997] Michael Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2, pages 93–177. Elsevier, 1997. [cited in page: 29]

[Muskens, 1996] Reinhard Muskens. Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy*, 19:143–186, 1996. [cited in page: 2, 42, 70]

[Muskens, 2001] Reinhard Muskens. Lambda grammars and the syntax-semantics interface. In R. van Rooy and M. Stokhof, editors, *Thirteenth Amsterdam Colloquium*, pages 150–155, 2001. [cited in page: 29, 30]

[Parigot, 1992] Michel Parigot. Lambda-my-calculus: An algorithmic interpretation of classical natural deduction. In *LPAR*, pages 190–201, 1992. [cited in page: 90]

[Partee, 1975] Barbara H. Partee. Montague grammar and transformational grammar. In *Linguistic Inquiry*, volume VI, pages 203–300, 1975. [cited in page: 22, 23, 27]

[Partee, 1997a] Barbara H. Partee. The development of formal semantics in linguistic theory. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, chapter 1, pages 11–38. Blackwell, 1997. [cited in page: 23, 29]

[Partee, 1997b] Barbara H. Partee. Montague grammar. In Johan van Benthem and Alice ter Meulen, editors, *The Handbook of Logic and Language*, chapter 1, pages 5–91. Elsevier, Amsterdam, 1997. [cited in page: 23, 27]

[Peirce, 1955] Charles Peirce. Abduction and induction. In *Philosophical writings of Peirce*. Dover Publications, 1955. [cited in page: 240]

[Peregrin, 2003] Jaroslav Peregrin. Introduction. In *Meaning: the Dynamic Turn*. Elsevier, Oxford, 2003. [cited in page: 43, 44]

[Plotkin, 1975] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theoretical Computer Science*, 1(2):125–159, 1975. [cited in page: 82]

[Pogodalla, 2004] Sylvain Pogodalla. Computing semantic representation: Towards ACG abstract terms as derivation trees. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pages 64–71, May 2004. [cited in page: 30]

[Russell, 1903] Bertrand Russell. *The Principles of Mathematics*. Cambridge University Press, 1903. [cited in page: 25]

[Scott and Strachey, 1971] Dana Scott and Christopher Strachey. Toward a mathematical semantics for computer languages. In *the Symposium on Computers and Automata*, volume XXI, pages 19–46. Polytechnic Institute of Brooklyn, April 1971. [cited in page: 81]

[Seuren, 1985] Pieter A. M. Seuren. *Discourse Semantics*. B. Blackwell, 1985. [cited in page: 54]

[Shan, 2002] Chung-chieh Shan. A continuation semantics of interrogatives that accounts for Baker's ambiguity. *CoRR*, 2002. [cited in page: 90]

[Shan, 2004] Chung-chieh Shan. Delimited continuations in natural language. In Hayo Thielecke, editor, *Proceedings of the Fourth ACM SIGPLAN Continuations Workshop (CW'04), Venice, Italy*, Birmingham B15 2TT, United Kingdom, January 2004. School of Computer Science, University of Birmingham. [cited in page: 90]

[Shan, 2005] Chung-chieh Shan. *Linguistic Side Effects*. PhD thesis, Harvard University, 2005. [cited in page: 43, 148]

[Stalnaker, 1970] Robert Stalnaker. Pragmatics. *Synthese*, 22(1-2):272–289, 1970. [cited in page: 44, 45]

[Stalnaker, 1973] Robert Stalnaker. Presuppositions. *Journal of Philosophical Logic*, 2:447–457, 1973. [cited in page: 54, 61, 63, 153]

[Stalnaker, 1974] Robert Stalnaker. Pragmatic presuppositions. In Milton Munitz and Peter Unger, editors, *Semantics and Philosophy*, pages 197–213. New York University Press, 1974. [cited in page: 54, 61, 63, 237]

[Stalnaker, 1978] Robert Stalnaker. Assertion. In Peter Cole, editor, *Syntax and Semantics*, volume 9, pages 315–332. Academic Press, 1978. [cited in page: 44, 45, 91]

[Stalnaker, 1999] Robert Stalnaker. *Context and Content: Essays on Intentionality in Speech and Thought*. Oxford University Press, 1999. [cited in page: 46]

[Strachey and Wadsworth, 1974] Christopher Strachey and Christopher P. Wadsworth. Continuations: A mathematical semantics for handling full jumps. Technical report, Oxford University, Computing Laboratory, 1974. [cited in page: 81]

[Strachey and Wadsworth, 2000] Christopher Strachey and Christopher P. Wadsworth. Continuations: A mathematical semantics for handling full jumps. *Higher-Order and Symbolic Computation*, 13(1):135–152, April 2000. [cited in page: 81]

[Strawson, 1950] Peter Frederick Strawson. On referring. *Mind*, 59(235):320–344, July 1950. [cited in page: 58]

[Strawson, 1952] Peter Frederick Strawson. *Introduction to Logical Theory*. New York, Wiley, 1952. [cited in page: 58, 59]

[Tarski, 1944] Alfred Tarski. The semantic conception of truth. *Philosophy and Phenomenological Research*, 4, 1944. [cited in page: 22]

[Tarski, 1956] Alfred Tarski. The concept of truth in formalized languages. In *Logic, Semantics, Metamathematics*, pages 152–278. Oxford Clarendon Press, 1956. [cited in page: 6]

[van Benthem, 1986] Johan van Benthem. *Essays in Logical Semantics*. D. Reidel Publishing Company, Dordrecht, Holland, 1986. [cited in page: 25]

[van Benthem, 1988] Johan van Benthem. The semantics of variety in categorial grammar. In W. Marciszewski W. Buszkowski and J. van Benthem, editors, *Categorial Grammars*. John Benjamins, 1988. [cited in page: 25]

[van der Sandt, 1988] Rob A. van der Sandt. *Context and Presupposition*. Croom Helm, New York, 1988. [cited in page: 54, 61]

[van der Sandt, 1992] Rob A. van der Sandt. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377, 1992. [cited in page: 59, 64, 66]

[van Eijck and Kamp, 1997] Jan van Eijck and Hans Kamp. Representing discourse in context. In Johan van Benthem and Alice ter Meulen, editors, *The Handbook of Logic and Language*, chapter 3, pages 179–237. Elsevier, Amsterdam, 1997. [cited in page: 65, 68, 69]

[van Eijck and Kamp, 2011] Jan van Eijck and Hans Kamp. Discourse representation in context. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 181–252. Elsevier, 2nd edition, 2011. [cited in page: 65, 68, 69]

[Vanderschraaf and Sillari, 2009] Peter Vanderschraaf and Giacomo Sillari. Common knowledge. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2009 edition, 2009. [cited in page: 120]

[von Heusinger, 1997] Klaus von Heusinger. Definite descriptions and choice functions. In Seiki Akama, editor, *Logic, Language and Computation*. Dordrecht, 1997. [cited in page: 48]

[Wittgenstein, 1921] Lugwig Wittgenstein. Logisch-philosophische abhandlung. In Wilhelm Ostwalds, editor, *Annalen der Naturphilosophie*. 1921. [cited in page: 45]

[Zeevat, 1989] Henk Zeevat. A compositional approach to discourse representation theory. *Linguistics and Philosophy*, 12(1):95–131, 1989. [cited in page: 70]

[Zeevat, 1999] Henk Zeevat. Demonstratives in discourse. *Journal of Semantics*, 16(4):279–313, 1999. [cited in page: 154]