Semantic Modeling of Textual Entailment:

Proof-Based Annotation in a Compositional Framework

Published by

LOT
Trans 10
3512 JK Utrecht
The Netherlands

phone: +31 30 253 6111
e-mail: lot@uu.nl
http://www.lotschool.nl

ISBN: 978-94-6093-189-5
NUR: 616

Semantic Modeling of Textual Entailment:

Proof-Based Annotation in a Compositional Framework

Semantische Modellering van Tekstuele Implicatie

Een Bewijs-Gebaseerde Benadering

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit Utrecht
op gezag van de rector magnificus, prof. dr. G.J. van der Zwaan,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op
maandag 26 oktober 2015
des middags te 12.45 uur

door

Assaf Toledo

geboren 17 september 1980 te Netanya, Israël

Promotoren:	Prof. dr. Y. S. Winter Prof. dr. M.B.H. Everaert

The research described in this thesis was supported by a VICI grant number 277-80-002 by the Netherlands Organisation for Scientific Research (NWO).

This dissertation is dedicated to my family and to Lada

Contents

List of Tables

List of Figures

Acknowledgements

In writing a thesis, one should be aware that most people will read only the acknowledgments. So important is this section that I procrastinated writing it till the last minute. Many, many people have been involved in my research, and even more have given me support and help. I can only hope I did not miss anyone here.

First of all, I would like to thank my supervisor Yoad Winter for his guidance and for helping me to find my way in this project. Many of the ideas presented in this thesis originated in his innovative mind. Yoad reviewed the numerous revisions of this thesis, and I am deeply grateful for his time and wise consideration. Yoad's care, help, and encouragement at the time when I was housebound were invaluable and are greatly appreciated. I would also like to thank Martin Everaert for being my first promoter during a greater part of these four years, for his sympathetic ear and for his advice. I also thank Martin for his help in the translation of the summary of the thesis to Dutch. Thank you to Joost Zwarts for reviewing the translation.

I wish to thank the members of the committee that reviewed my manuscript, offering valuable remarks and suggestions: Johan Bos, Alexis Dimitriadis, Jan Odijk, Dan Roth and Khalil Sima'an. I am especially grateful to Alexis for the many talks we had in the course of the project and for his comments and ideas.

I had the pleasure to collaborate with Sophia Katrenko, whose intelligence and know-how I admire, and who has by and by become a good friend. So has Marijn Struiksma, who was always there for me when I needed advice or simply a sympathetic ear. I also want to thank Choonkyu Lee, Eva Poortman and Hanna de Vries for being good friends and helping anytime I needed something.

I could not have accomplished this work on my own: It was made possible thanks to its embedding within the VICI research project between logic and common sense at the Utrecht Institute of Linguistics (UiL OTS). I wish to thank my colleagues in the research team, who invested much time and effort in this project, practically from the very beginning and until the last day. My sincere thanks go to Heidi Klockmann and Stavroula Alexandropoulou, whose

dedicated annotation work resulted in SemAnTE 1.0, the first of the corpora released. This initial phase of the research helped us to envision and plan its more advanced stages. I am likewise grateful to the rest of our committed annotation team, whose effort resulted in the compilation of the SemAnTE 2.0 corpus: Elena Callegari, Sophie Chesney, Robert Grimm, Kyriaki Neophytou, Anton Nguyen and Tom Roberts. This task could not have been accomplished without their dedication and excellent teamwork.

I am immensely grateful to Pepijn Kokke for investing a tremendous effort in the development of the SemAnTE annotation platform. As one of the team programmers, Pepijn took part in the majority of decisions about the system architecture. His work was critical in developing some of the system's most important components: the lexicon, the transformation of an annotated tree to a logical formula, and the typed-lambda calculus engine. He spent uncountable hours on this project and his commitment has been truly remarkable. I learned a lot from Pepijn's skills and creative ideas.

I appreciate Benno Kruit's substantial contribution to our annotation platform. Benno developed the interface between the system and a theorem prover and proved the soundness and incompleteness of the system. I appreciate his generosity in allowing me to include his proof in this thesis's appendix.

I also want to thank Ido Dagan and Asher Stern from Bar-Ilan University, for collaborating with me in the earlier stages of my research and for giving me feedback throughout the entire project. And thank you, Ido, for coining the trade-name *Annotating-By-Proving* for the methodology that we developed.

Thank you to Maaike Schoorlemmer for being there when I needed an ear, for helping me with formal procedures, and for keeping an eye on my progress.

Thank you to Henriëtte de Swart for listening to me and giving me kind advice when I needed it most.

I also want to thank the secretaries of UiL OTS – Yvonne van Adrichem and Maritte Bonenkamp, who helped me in my first steps in the institute, and Martine Paulissen.

This is yet another opportunity for me to thank my teachers from the Hebrew University of Jerusalem – Edit Doron, Malka Rappaport Hovav – for introducing me to the world of linguistics, for teaching me so much, and for recommending me to enter the PhD program at Utrecht University.

I am grateful to Nina Luskin for investing many hours in editing and correcting my English in a substantial part of this thesis and for debating with me various linguistic issues. It should be fairly easy for the reader to tell which sections Nina did not work on.

On a more personal note, I want to thank my officemates, Desiree Capel, Marjolein van Egmond and Anna Sara Hexeberg Romøren. We had such a wonderful rapport that it saddens me to have to speak about Room 0.45 in the past tense. Anna Sara supported me when I was housebound; Marjolein helped me to find an apartment (and transported my stuff to the airport – although that's not such a nice memory); and Desiree was always there when I needed a sympathetic ear and reassurance. I will remember warmly Paula Cox, who spent quite a few hours with us and whose charming personality enlivened our work. Special thanks to Anna Sara and Marjolein for being my paranymphs and to Desiree for translating the summary of this thesis to Dutch.

I also want to mention Briggita Keij, Liquan Liu, and Nadya Goldberg, who became my close friends over the four years, and whose friendship I greatly appreciate. It was always a pleasure to be in your company. An especial thank you to you, Nadya, for retouching the cover picture of this thesis.

When I first entered UiL OTS, I was astounded by the sheer size of the place and did not realize that I was going to meet there so many great minds from all over the world. It was an exciting experience for me to work side by side with you: Ana Aguilar Guevara, Daria Bahtina-Jantsikene, Dominique Blok, Lisa Bylinina, Marta Castella, Anna Chernilovskaya, Xiaoli Dong, Jakub Dotlačil, Anja Goldschmidt, Ileana Grama, Alexia Guerra Rivera, Mirjam Hachem, Carolien van den Hazelkamp, Lotte Hendriks, Marko Hladnik, Lena Karvovskaya, Loes Koring, Bert Le Bruyn, Anne van Leeuwen, Tom Lentz, Zenghui Liu, Sophia Manika, Yaron McNabb, Mike Olson, Sandrien van Ommen, Liv Persson, Anne-France Pinget, Erin Pretorius, Jolien Scholten, Maartje Schulpen, Marko Simonovic, Björn 't Hart, Anna Volkova, Franca Wesseling, Jingwei Zhang and Rob Zwitserlood.

Living abroad for a long period of time is never easy, and it was essential for me to keep in touch with people from back home. I would like to mention long-time friends from Israel who came to visit me in Utrecht and whose friendship I cherish: Naama Ben Arie and Orly Zvitia, Einat Brod and Michael Katkov, Micha Yochanan Breakstone, Keren Fleshler, Elad Granot, Ori Rapaport and Shelly Dar Rapaport, and Yael Leyden van Amstel. My other Israeli friends also helped me to feel connected to my home country: Galit Agmon, Rakefet Rebecca Ben Ari, Eva Kimel, Royi, Malka and Eli Glass, Ilona Spector Shirtz, Avigail Tsirkin-Sadan, Viki Vinokur, and Yishay Weiss.

My strength and motivation for achievement are due to my family. I want to thank my parents, Rivka and Yehuda; my brother Ron; my sister Chen; and my grandmother Rivka. Your unwavering support has been invaluable, sustaining me and providing inspiration.

Last but not least, I want to thank my beloved girlfriend Lada Karmazin. Your love and encouragement during these four years gave me heart – never mind the many sad goodbyes at different airports. Thank you for your willingness to embark on this long journey with me and for your patience and endurance along the way. Now that the goal is accomplished, I hope you, too, feel rewarded.

CHAPTER 1

Introduction

This thesis explores the use of formal semantic theory in explaining textual entailments and integrates it in a proof-based computational framework of natural-language inferences.

The proposed semantic model incorporates a typed lexicon that encodes some of the most common inferential phenomena in the Recognizing Textual Entailment (RTE) corpora: appositive, intersective and restrictive modification, as well as simple existential and universal quantification. In a given entailment pair, the words of the premise and the conclusion are assigned denotations by being bound to lexical labels. Sentences are analyzed syntactically and interpreted according to the principle of compositionality. Entailments are formally described by applying a truth-conditionality criterion to the semantic terms calculated for the premise and conclusion. This analysis allows us to define a semantic model of entailment based on simple syntactic and lexical models.

The thesis describes an annotation platform that was developed on the basis of the semantic model. This platform is used for creating and analyzing new entailment data. As part of this process, both the model and the platform were tested and fine-tuned. The platform integrates a standard stochastic parser, a part-of-speech tagger, a lambda calculus engine and a first-order theorem prover with a graphical user interface. It is used in a semi-automatic workflow, as human intervention is required for fixing parsing errors and correcting or completing the semantic annotation of the premise and the conclusion. The corrections and completions are performed through the user interface. This workflow allows human annotators to ascertain that the phrase-based syntactic analysis and the semantic annotations lead to a formal proof of the entailment. The platform is logically sound with respect to the semantic theory, and therefore, finding a proof from a premise to a conclusion indicates that the semantic theory formally accounts for the entailment.

The thesis reports results of using the platform for creating and annotating entailment data explained by the semantic theory. The platform was utilized to compose positive (entailing) and negative (non-entailing) examples, which were compiled in a new corpus of 600 annotated entailments.

The rest of this introduction describes the concept of textual entailment, explains the utility of its automatic recognition, and lays out the goals of this thesis and its research questions. Additionally, it elaborates some central aspects of the semantic model, its implementation by an annotation platform and the methodologies employed for composing and annotating entailment data.

Textual Entailment in natural language is a relation between two units of text – a premise referred to as "the text" t and a hypothesis h. Intuitively, an entailment relation means that the information provided by h follows from the information provided by t. For example:

(1) *t*: The largest search engine on the web, Google, receives over 200 million queries each day through its various services.

h: Google operates on the web.

Entailments are indicated by the notation: $t \Rightarrow h$.

Recognizing entailment can be useful for a number of tasks in Natural Language Processing (NLP), such as Searching, Question-Answering, Multi-Document Summarization and Machine Translation (Dagan et al., 2006, 2013, Sammons et al., 2010). For instance, an entailment recognizer can widen the coverage of a search engine by allowing it to identify documents that entail a sentence and not merely contain some of its words. Thus, a recognizer makes it possible to cast a search task as an entailment recognition task. It can also serve as an inference engine embedded in a more complex system. For example, multi-document summarization algorithms can employ a recognizer for identifying candidate sentences which are entailed by others and are therefore redundant. The interest in automatic entailment recognition found expression in the launching of the RTE challenges (Dagan et al., 2006). The aim was to assess the performance of state-of-the-art entailment recognizers, which also involved the compilation of entailment corpora (Bar-Haim et al., 2006, Giampiccolo et al., 2007, 2008, Bentivogli et al., 2009, 2010b, 2011).

The appeal of the RTE paradigm is that it sets the bar for the performance of entailment recognizers according to the demands of NLP tasks that recognizers aim to be embedded in. However, targeting a wide coverage of natural texts leads to a big challenge in developing accurate entailment recognizers for these data. We propose that this challenge can be at least partially tackled by using an explicit linguistic theory that accounts for the input data

This thesis proposes a way to develop a theory-based paradigm for investigating textual entailment. The paradigm involves two steps. The first is to develop tools for creating a corpus of entailment data that can be encompassed by an explicit linguistic theory. The second is to use the theory and the corpus for developing and improving entailment recognizers that classify entailing and non-entailing pairs. This paradigm is premised on the assumption that entailment recognizers could be made more accurate if an explicit linguistic theory explains at least some of the data that they are designed to cover. The thesis focuses only on the initial step in materializing this paradigm: developing a computational framework that enables an annotation methodology for obtaining theory-based entailment data.

The first question that this thesis investigates is: to which linguistic phenomena is it feasible and expedient to apply a theory-based paradigm? Our initial assumption was that inferences stemming from well-analyzed semantic phenomena are the most suitable candidates for being explored in the proposed approach. Specifically, we focus on the phenomena of *apposition* and *restrictive modification.* We hypothesized that these semantic phenomena are common in natural language entailment and used the RTE corpora to examine this hypothesis. The focus is on the first four RTE corpora, which contain short entailment pairs as illustrated in (1), whereby each pair is assigned an entailing/non-entailing label. This example contains the apposition *The largest search engine on the web, Google*, which allows us to infer that the largest search engine on the web is Google. An instance of restrictive modification appears in the phrase *the [largest [search engine on the web]]*, which, together with the semantics of definite articles, licenses the inference that Google is a search engine on the web.

Building on previous work by Van Strien (2009), an annotation project was carried out that probes the frequency of inferences stemming from modification in the entailing pairs of RTE 1–4. In this annotation work, two human annotators marked all appositive and restrictive modifications that played a role in their interpretation of the entailment relation between the text and hypothesis. Analyzing the annotations indicated that inferences triggered by appositive and restrictive modification are found in 80% of the positive pairs of these corpora. This work was released as a free corpus, entitled Semantic Annotation of Textual Entailment (SemAnTE) 1.0. It adds to other annotation projects conducted on the RTE datasets (Garoufi, 2007, Bentivogli et al., 2010a, Sammons et al., 2010) and leads directly to the main effort of this research: developing a theory-based framework for analyzing textual entailment.

The process of compiling SemAnTE 1.0 revealed challenges in annotating inference-triggering phenomena and verifying the accuracy of the annotations marked. The annotation guidelines instruct annotators to describe entailments informally first, and only then annotate the linguistic phenomena involved in their interpretation. To do that, annotators informally analyzed the inference from the text to the hypothesis and then marked the instances of modification phenomena that occur in their analysis. Since this analysis is informal and involved translating an intuitive entailment judgment to a sequence of concrete linguistic phenomena, an annotator may miss one or more such structure or annotate it inaccurately. In addition, it transpired that even a review by a second annotator may not detect all errors.

Furthermore, for the purpose of this work, inter-annotator agreement (IAA) checks did not prove to be an optimal tool for detecting annotation errors. Indeed, in many examples there is a number of valid ways to analyze an entailment pair, so it is difficult to single out one "correct" annotation. For instance, as explained above in respect of (1), the semantics of restrictive modification and definite articles can be invoked to infer that Google is a search engine on the web. Consequently, the inference that Google searches on the web can be based on the nominalization relation between the verb *search* and the noun *search*, while the inference that Google operates on the web rests on a lexical rule. Alternatively, another interpretation of the entailment can be anchored in the information that Google receives queries through its various services. In this path, using world knowledge, we infer that Google receives queries through its web services, and that, therefore, it operates on the web. Thus, divergent annotations may occur simply because annotators follow different legitimate inferential paths in recognizing an entailment. As a result, IAA checks may show disagreement between annotators when their annotations are in fact valid.

It follows that, without a precise model of entailment, the accuracy of annotations has to be examined manually, on a pair to pair basis, and in light with an annotator's subjective interpretation of each entailment. This makes the task labor intensive and error-prone.

A theory-based paradigm for investigating textual entailment provides a solution for these problems in using an informal annotation scheme. This is because a formal model can be expedient for ascertaining that the annotations explain the entailment or lack thereof.

As a basic framework for such a model, we adopted state-of-the-art formal semantics. This leads to the main research question of this thesis: how can formal semantic theory be implemented in an annotation platform that would facilitate the modeling of textual entailment. A crucial consideration was to render the platform sound with respect to the semantic theory; that is, finding a deduction from the text to the hypothesis must formally imply that the text logically entails the hypothesis within the semantic theory.

This rationale has led to a methodological principle termed *Annotating-By-Proving* (ABP). This procedure dictates that an entailing pair is considered well-annotated only if the marked annotations allow the platform to generate a proof. This proof shows that the analysis of the text based on the implemented theory entails the analysis of the hypothesis. An extension of ABP covers non-entailing pairs as well.

ABP requires that entailment relations and lack thereof are explained formally based on an explicit linguistic theory. We examined the coverage of this requirement by creating theory-based entailment data – new entailing and non-entailing pairs that formally demonstrate the inferences which a given semantic theory accounts for. For creating these data, we developed an annotation platform that allows annotators to employ ABP. The platform relies on a formal system in ascertaining that the annotations lead to a deductive process from the text to the hypothesis. In practice, if the system is unable to generate a proof based on the annotations, this is likely due to a missing or an inaccurate annotation. Moreover, in this way annotation errors are spotted and corrected in the course of the annotation process. ABP also allows the annotator to mark linguistic phenomena according to the subjective inferential path that she posits based on her interpretation of the data. Although informal self-checks by an annotator are still required, this approach minimizes the load of deep annotation checks.

Special attention is given to the simplicity of the representations employed by the semantic theory. It is assumed that, the simpler the representations are, the more easily they can be learned by a machine. To obtain simplicity, as little information as possible is added to the syntactic structures generated by contemporary phrase-structure parsers. These considerations make a formal compositional semantic theory a natural candidate as the theory guiding the semantic annotation of entailment.

Formal semantic theories frame entailment as a primary linguistic phenomenon. As put by Richard Montague, "The basic aim of semantics is to characterize the notion of a true sentence (under a given interpretation) and of entailment" (Montague, 1970). This reasoning directed the development of semantic theories, rendering them suitable frameworks for capturing basic principles of natural-language entailments. At the same time, compositional theories allow us to assume very simple representations. In this thesis we obtain this simplicity by limiting ourselves to a basic markup language, with notations placed above standard phrase structures.

For all that, current formal semantic theories are not easily applicable to real-life texts. First, semantic theories tend to analyze linguistic phenomena in isolation and only in relatively restricted syntactic environments. Second, inferences in natural language stem from an intricate combination of inferential phenomena, only some of which are covered by formal semantics. In particular, many aspects of common-sense reasoning are beyond the realm of standard logical approaches in formal semantics. Thus, it is evident that a theory-based paradigm for investigating textual entailment suffers from coverage problems. However, without a thorough investigation of the limits of current semantic theories, it is hard to develop modular systems that would benefit from their insights.

Accordingly, a large part of the work in this thesis aims to check the boundaries of current semantic theory in the formal analysis of entailment data. This examination is performed in three steps: (1) defining a semantic model of entailment by adapting a standard semantic application of Tarski's truth-conditionality criterion and representing a set of inferential phenomena rooted in modification and simple quantification; (2) developing a proof-based annotation platform that implements the semantic model using an off-the-shelf parser, a part-of-speech tagger and a theorem-prover; and (3) using the platform to compose entailing and non-entailing examples whose inferential paths can be explained by the semantic model. This approach allows to begin with a theory that models a small set of linguistic phenomena and then augment it to cover increasingly complex syntactic constructions and semantic inferences.

The process of annotation using the platform is semi-automatic. A syntactic analysis is performed by Klein and Manning's stochastic parser (2003) and Toutanova et al.'s part of speech tagger (2003). Next, based on their intuitive interpretation of the sentences, human annotators fix parsing errors and correct or complete the annotations of linguistic phenomena that either license or block entailment. Finally, the annotation platform interfaces a first-order theorem prover to automatically search for an inferential process from the semantic term calculated for the text to the one calculated for the hypothesis. In this way the platform implements a model of entailment that connects computational models of syntax, semantics and logic. It allows human annotators without training in logic to verify the existence of formal deductions.

Standard formal semantic analyses for modeling entailment involve the use of higher-order logic for modelling semantic phenomena. This poses a challenge for the annotation platform, since searching for a proof in a higher-order logical framework is computationally expensive. For this reason, prior to theorem proving, the platform lowers all higher-order terms to first order using a simple heuristic, which has shown itself to be sufficiently general to preserve all inferences analyzed in this work. Proving first-order logic inferences is performed by means of McCune's (2010) *Prover9* theorem prover.

The annotation platform was used to compile a new theory-based corpus, which has been publicly released as part of this work. The first part of this corpus contains entailing pairs created based on RTE examples which were simplified to avoid un-modeled inferential phenomena or unsupported syntax. All entailing pairs in the corpus are treated using the annotating-by-proving methodology, and consequently, they are construed as logical entailments in the semantic theory. The second part of the corpus consists of couplets composed of entailing and non-entailing pairs. The couplets are constructed so as to show a minimal contrast sufficient for producing opposite entailment judgements accounted for by the semantic model. This setup demonstrates the predictive power of the model and the fragment of English to which the model is applicable. The corpus contains 600 entailment pairs in an entailing/non-entailing ratio of 2:1. It is entitled SemAnTE 2.0 and it is freely available on the web.[1]

This thesis is organized as follows. Chapter 2 provides an overview of the Recognizing Textual Entailment task and reviews relevant literature. Chapter 3 describes the linguistic phenomena annotated in this study and illustrates them with examples from the RTE corpora. Chapter 4 presents a formal model of entailment adopted from a standard semantic theory and its use in analyzing the inference in some RTE examples. The annotation platform that implements this semantic model is described in Chapter 5 focusing on its architecture and components. Chapter 6 details the compilation of the corpus by using the platform and describes possible future developments. Chapter 7 con-

[1] See: `http://logiccommonsense.wp.hum.uu.nl/resources`

cludes. Appendix A presents a previously obtained proof that the annotation platform is sound even if incomplete on account of the order-lowering algorithm employed for converting higher-order terms to first-order. Appendix B describes the format in which the resources produced by this work, SemAnTE 1.0 and SemAnTE 2.0, are released. Appendix C illustrates a use case of SemAnTE 1.0. Appendix D provides references to RTE examples that were used for illustrations throughout this thesis.

CHAPTER 2

The Recognizing Textual Entailment Task

Overview

This chapter describes the Recognizing Textual Entailment (RTE) task originally proposed by Dagan and Glickman (2004). It opens by elucidating the motivation for and the goal of the task. Next, it describes the corpora created for the RTE challenges, with several examples. A number of approaches to entailment recognition are elaborated next, illustrated by describing existing entailment recognizers. The chapter concludes with a discussion of previous annotation works on the RTE.

2.1 Motivation

In the field of Natural Language Processing (NLP), the concept of inference is taken to be “the process of concluding the truth of a textual statement based on (the truth of) another given piece of text” (Dagan et al., 2013, p. 23). Recognizing inference can be useful for a number of various NLP applications, including Searching, Question-Answering, Information Extraction, Information Retrieval and Multi-Document Summarization. It has been noted that “many semantic inference needs in NLP can be cast in terms of textual entailment” (Dagan et al., 2013, p. 24). A growing interest in recognizing inference in free text has thus evolved into the development of a textual entailment paradigm, as “a unifying framework for applied semantic inference”.

2.2 Challenges and Corpora

The seven RTE challenges conducted between 2006 and 2011 sought to assess the performance of state-of-the-art entailment recognizers. To this end, they defined the latest computational paradigm for compiling entailment corpora (Dagan et al., 2006, Bar-Haim et al., 2006, Giampiccolo et al., 2007, 2008, Bentivogli et al., 2009, 2010b, 2011). Such corpora were prepared accordingly, to provide data sets for training and evaluating recognizers.

The first five challenges focused mostly on the entailment recognition setup at the sentence level, illustrated in Example (1) and repeated below in (2). The sixth and seventh challenges, which targeted a more complex setup that requires tools for discourse analysis, are beyond the scope of this thesis.

The content of the corpora was largely shaped by the need for recognizing inferences in many of the NLP applications, which had motivated the RTE task in the first place. Therefore, the pairs included in the corpora are either copied or adapted from the datasets used in these applications. For instance, (2) is a paraphrase of a relation extraction task. This pair, which comes from Pair 955 in RTE 4 Test set, illustrates a textual entailment setup for the extraction of an *operates on* relation between *Google* and *the web* from the given *t*. Such a relation extraction task, then, can be performed by an Information Extraction system using a recognizer. In other words, the recognizer, which bundles a range of inference components and knowledge resources, can be effectively used as an embedded technology in other applications.

(2) *t*: The largest search engine on the web, Google, receives over 200 million queries each day through its various services.
h: Google operates on the web.

As the textual-entailment paradigm targets natural inferences, a guideline for annotating pairs as either entailing (positive) or non-entailing (negative) must be intuitively plausible. That is, a pair is marked as positive if *humans reading t would typically infer that h is most likely true* (Dagan et al., 2006, p. 2). Such a formulation captures the perception of entailment as a unidirectional relation between texts, which is in line with its formal semantic definition, requiring *h* to be true in every possible world where *t* is true (Gamut, 1991). However, this formulation relies on vague, commonsensical terms such as *typically* and *most likely true* rather than formal concepts such as *truth*, *universal quantification* and *possible worlds.* As an example, see (3) below. This pair was annotated as a positive entailment, which means that the RTE annotators considered it an intuitive inference. Yet, logically the inference does not hold. To begin with, the statement in *t* is a subjective opinion, and as such can turn out to be empirically wrong. Furthermore, the message is heavily hedged by the qualifiers *may be* and *apparently.* In this case, the annotators' judgment may have been swayed by the authority of Harvard, which lent the message credibility.

(3) *t*: Researchers at the Harvard School of Public Health say that people who drink coffee may be doing a lot more than keeping themselves awake – this kind of consumption apparently also can help reduce the risk of diseases.

h: Coffee drinking has health benefits.

The agreement checks conducted as part of the corpora preparation indicated that, in spite of the non-formal nature of the guideline adopted, the judgements of the RTE annotators on the validity of entailments were for the most part consistent.

Table 2.1 shows the number of pairs in the data sets of the first five challenges.[1] Originally, the pairs included in RTE 1-3 were categorized in two classes (*yes*- and *no-entailment*), while in RTE 4-5 in three (*entailment*, *contradiction* and *unknown*: not entailing but also not contradicting). de Marneffe et al. (2008) introduced the contradiction annotation for the *non-entailing* pairs in RTE 1-3 as well, thus subjecting all of the first five RTE corpora to a three-way categorization. The guideline for marking a pair as a contradiction is based on whether *a human reader would say that h is highly unlikely to be true given the information described in t* (de Marneffe et al., 2008). The percentage of positive and negative entailment pairs is balanced fifty-fifty.

Challenge	Data set	Pairs count
RTE-1	Development	567
	Test	800
RTE-2	Development	800
	Test	800
RTE-3	Development	800
	Test	800
RTE-4	Test	1000
RTE-5	Development	600
	Test	600

Table 2.1: The datasets of RTE-1–5

Examples of RTE pairs that this thesis uses for illustration purposes are collected in Table 2.2. For each example, the entailment judgement is indicated, as well as the NLP application associated with it: Question-Answering (QA), Information Extraction (IE), Information Retrieval (IR) or Multi-Document Summerization (SUM). Appendix D lists all RTE pairs referenced in this thesis. For works that employ a textual entailment recognizer in Question-Answering systems see Harabagiu and Hickl (2006), Celikyilmaz et al. (2009) and Ferrández et al. (2011); for works that use a recognizer in Relation Extraction systems

[1] The forth challenge provided only testing data; the training was done based on the datasets from the previous challenge.

see Romano et al. (2006), Bar-Haim et al. (2007a) and Roth et al. (2009); and for a utilization of the entailment recognizer of Hickl et al. (2005) in a Multi-Document Summarization system see Harabagiu et al. (2007). The studies of Bar-Haim et al. (2007a) and Roth et al. (2009) also set out to explore the usage of an entailment recognizer in a search setup. In addition, see Marelli et al. (2014) for a more recent textual entailment task as part of SemEval 2014.

2.3 Entailment Recognizers

Entailment recognizers rely on several different approaches to the computational modeling of entailment. This section surveys some of these and elaborates on specific systems that are relevant to this thesis.

Similarity-based Approaches

One approach to entailment relies on measures of similarity between the text and the hypothesis. The underlying assumption is that positive entailment pairs are more likely to have similar text and hypothesis compared to negative pairs. A number of similarity measures have been proposed to model this intuition for the purposes of entailment prediction.

The simplest similarity-based approach appeals to the lexical overlap between the text and the hypothesis. The tokens of the text and the hypothesis are lemmatized and a prediction is made based on their overlap (Mehdad and Magnini, 2009). More advanced measures use knowledge bases such as WordNet (Fellbaum, 1998) to identify synonymy and hyponymy relations between tokens in the text and in the hypothesis (Adams et al., 2007), which count as an overlap.

Yet another similarity-based approach is anchored in measures of syntactic-overlap, e.g., between dependency structures (de Marneffe and Manning, 2008) and between standard constituency structures. These methods are studied and compared to the lexical-similarity based approach in Katrenko and Toledo (2011). A dependency-overlap measure is obtained by parsing the text and hypothesis using a dependency parser and then calculating the extent to which dependency relations (triplets comprising the name of a relation and the arguments) that constitute the hypothesis tree appear in the text tree. A constituency-overlap measure is calculated similarly, based on phrases in phrasal structures of the text and the hypothesis. Decision-tree classifiers based on these measures, as well as on a simple lexical measure, were trained on RTE 1-5 development sets and evaluated based on the corresponding test sets. The results are reported in Table 2.3 in terms of recognition accuracy the number of correct predictions divided by the number of pairs in the test. These results suggest that lexical-overlap is the strongest baseline, followed by dependency-overlap, and then constituency-overlap. A small increase in accuracy is usually

Dataset	Pair	Text	Hypothesis	Task	Ent.
RTE 1 Dev.	19	Researchers at the Harvard School of Public Health say that people who drink coffee may be doing a lot more than keeping themselves awake – this kind of consumption apparently also can help reduce the risk of diseases.	Coffee drinking has health benefits.	IR	Yes
RTE 2 Dev	611	The book contains short stories by the famous Bulgarian writer Nikolai Haitov	Nikolai Haitov is a writer.	QA	Yes
RTE 2 Test	410	The head of the Italian opposition, Romano Prodi, was the last president of the European Commission.	Romano Prodi is a former president of the European Commission.	QA	Yes
RTE 3 Dev.	606	Amsterdam police said Wednesday that they have recovered stolen lithographs by the late U.S. pop artist Andy Warhol worth more than $1 million.	Police recovered 81 Andy Warhol lithographs.	Sum	No
RTE 4 Test	862	Carole James is a newcomer in the leadership role of the NDP. She was elected in 2003 and aims to restore the party to power after they suffered an embarrassing defeat. In 2001, the NDP went from government to opposition with only two seats.	NDP won the 2001 election.	IE	No
RTE 4 Test	955	The largest search engine on the web, Google, receives over 200 million queries each day through its various services.	Google operates on the web.	IE	Yes

Table 2.2: Entailment Pairs from RTE 1–4

obtained by combining the lexical, dependency and constituency measures as features in a Bayesian Logistic Regression (see the *BLR* column). The *Base* column displays Mehdad and Magnini's results (2009).

Dataset	Lexical	Dep.	Const.	BLR	Base
RTE-1	**55.00**	54.38	50.00	54.63	52.25
RTE-2	58.50	58.75	56.63	**60.25**	55.87
RTE-3	62.63	58.63	56.25	**62.88**	61.5
RTE-4	57.30	55.60	50.70	**59.00**	58.7
RTE-5	57.00	53.00	53.33	**60.20**	57.08

Table 2.3: Accuracy of Lexical- and Syntactic-Overlap Baselines

The simplicity of similarity-based approaches such as lexical-overlap has rendered them a natural baseline for evaluating the performance of more sophisticated recognizers..

Formal Logic Approaches

In formal logic approaches, the text and hypothesis are translated into logical forms and then a theorem prover searches for a proof that deduces the representation of the hypothesis from that of the text. Systems created in this architecture are generally precision-oriented at the expense of recall. This is because a proof, once it is found, is a highly reliable indicator of an entailment relation. However, a proof is often not found due to partial or inaccurate descriptions of the text and hypothesis in logical forms or the deficient representation of the world knowledge. In such cases, the data might be inadequate for a reasonable prediction of the entailment status hence the low recall. A number of RTE systems designed in this general architecture address these obstacles.

The logical representation in Raina et al. (2005) is based on dependency structures, and so is that in Harabagiu et al. (2000). To make the system robust, world knowledge is learned automatically from resources such as WordNet and integrated by means of weighted abductive assumptions estimated by a cost model according to their likelihood. A likely assumption is assigned a lower cost than an unlikely one. Following Hobbs et al. (1993), a proof with a minimum cost is found using an abductive theorem prover. If a proof with a low cost is found, the system predicts that the entailment pair is positive. This system reached a recognition accuracy of 57% on RTE 1.

Bos and Markert's (2005) system utilizes a CCG parser (Bos et al., 2004) to represent the text and the hypothesis in discourse representation structures (DRS) that encapsulate information on argument structure, semantic roles, polarity, etc. (Kamp and Reyle, 1993). The DRSs of the text and the hypothesis are then translated into formulae in first order logic, and a theorem prover is invoked. Running with a strict theorem prover (Riazanov and Voronkov, 2002), the system reached a relatively high precision score of 76.7% in recognizing the positive cases in RTE 2 but its recall was low at 5.8%. The accuracy in this configuration amounted to 52%. In order to increase the recall, the system was extended with a heuristic based on a model builder. The rationale is that, if the text entails the hypothesis, then a model that satisfies the text is supposed to be of comparable dimensions to one that satisfies both the text and the hypothesis. Otherwise, it is likely that the text does not entail the hypothesis. This heuristic, which approximates the entailment, reached a recall of 73.5%. A hybrid approach that combines both methods, and also takes into account the task associated with the data (i.e. QA vs IE), reached an accuracy of 61.2%. A more recent discussion on different ways of producing world knowledge in order to increase recall can be found in Bos (2013).

Garrette et al. (2013) proposed a method to recast first-order semantics into probabilistic models of distribution, in line with Markov Logic Networks (Richardson and Domingos, 2006). Such a strategy allows converting candidate lexical relations between words in the text and in the hypothesis into weighted inference rules that are added to the logical form. This produces a weighted inference between the text and the hypothesis. Garrette et al. connect between Bos and Markert's (2005) formal approach, which relies on precise representations and strict theorem proving, on the one hand, and distributional approaches that capture graded similarity between concepts, on the other. Beltagy et al. (2013) extends this work by adding phrase-based inference rules and evaluating the framework's performance using the dataset of RTE 1. The accuracy of the system is 57%, which is higher than that obtained by Bos and Markert (2005) in their logic-only model (52%).

Transformational Approaches

Transformational approaches use rules that transform the representation of the text to one that contains the representation of the hypothesis. The representations are in the form of a dependency tree annotated with part-of-speech tags. This design was first introduced by Bar-Haim et al. (2007b; for a detailed account, see Bar-Haim, 2010) and generalized by Stern and Dagan (2011).

Stern and Dagan's implementation of this technique in the Bar Ilan University Textual Entailment Engine (BIUTEE) employs a set of hand-crafted syntactic rules (Lotan, 2012) that specify syntactic transformations e.g., *Genitive: Substitute an "X's Y" construction with a "the Y of X" construction.* To extend the coverage of its syntactic rule base, BIUTEE utilizes lexical and lexical-syntactic rules learned from knowledge resources such as, inter alia, WordNet (Fellbaum, 1998, Miller, 1995), FrameNet (Baker et al., 1998), VerbOcean (Chklovski and Pantel, 2004), Catvar (Habash and Dorr, 2003), Lin Similarity (Lin, 1998), and DIRT (Lin and Pantel, 2001). These rules are supplemented by a set of on-the-fly text manipulation operations such as addition/movement/replacement of words.

It is assumed that, in positive entailments, the text can be modified so as to contain the hypothesis through minimal changes. By contrast, in negative entailments, such an operation would be much more involved. In other words, in positive entailments fewer modifications are likely to be needed to render t representationally superordinate to h, and these changes are less fraught than in negative pairs. For example, paraphrasing the genitive "X's Y" as "the Y of X" is, in this sense, safer - or involves a lesser cost - than deleting or adding a negator. A cost model learns this confidence estimation process automatically based on training data. The polarity of the entailment relation (positive vs. negative) is predicted based on these estimations. The system was tested on the data sets of RTE 1, 2, 3 and 5, reaching an accuracy of 57.13%, 61.63%, 67.13% and 63.50%, respectively. To ascertain that, in modifying the text to incorporate the hypothesis, BIUTEE applies linguistically motivated rules, we conducted a pilot study, which is described in detail in Chapter 3 (for full results, see Appendix C).

Other Approaches

MacCartney and Manning's (2007) system recognizes monotonic relations (or lack thereof) between aligned lexical items in the text and hypothesis, and employs a model of compositional semantics to calculate a sentence-level entailment prediction. The recognition of monotonic relations is done using an adapted version of Sánchez Valencia's (1991) framework of Natural Logic. The alignment between the text and hypothesis is done based on a cost function that extends the Levenshtein string-edit algorithm, and the entailment is classified by a decision tree classifier, trained on a small data set of 69 handmade problems. The system was tested on RTE 3 and achieved relatively high precision scores of 76.39% and 68.06% on the positive cases in the development and test sets, respectively. However, it suffered from low recall scores of 26.70% and 31.71%, respectively.

Recognizers Performance

Table 2.4 lists the number of participants in the RTE 1–5 challenges and, for each of the latter, presents the performance of the most and the least accurate recognizer, as well as an average of all systems and the lexical-overlap baseline reported in Mehdad and Magnini (2009). The data are based on Kouylekov et al. (2011). The results show that the simple lexical-overlap measure emerged as a strong baseline for the RTE task.

Challenge	Participants	Best	Worst	Average	Base
RTE1	15	58.60	49.50	54.40	52.25
RTE2	23	75.38	52.88	59.77	55.87
RTE3	26	80.00	49.63	62.37	61.5
RTE4	26	74.60	51.60	59.35	58.7
RTE5	20	73.50	50.00	61.41	57.08

Table 2.4: Recognition Accuracy in RTE Challenges

2.4 The FraCaS Test Suite

The FraCaS test suite (Cooper et al., 1996) is an inference test set for evaluating the inferential competence of NLP systems and semantic theories. This evaluation is done by examining in which tests a system, or a theory, passes and in which ones it fails. It includes about 350 tests classified according to semantic and other linguistic phenomena such as *Generalized Quantifiers*, *Plurals*, *Anaphora* etc. The tests are set as triplets containing (a) an input T of one or more sentences, (b) a question S, and (c) a label indicating whether S should be answered positively, negatively, or cannot be answered given T (yes/no/unknown). The input T and the question S are both expressed in natural language as the test suite does not incorporate any formal semantic notation.[2] Example (4), which was taken from the *Plural* test group (id. 3.81), shows a question that should be answered positively; Example (5), taken from the *Anaphora* group (id. 3.119), shows a question that shoul be answered negatively; and Example (6), taken from the *Ellipsis* group (id. 3.143), shows a question that cannot be answered based on the given information.

(4) *T* Smith, Jones and Anderson signed a contract.
S Did Jones signed a contract?
L Yes

(5) *T* No student used her workstation. Mary is a student.
S Did Mary use her workstation?
L No

(6) *T* John spoke to Mary. So did Bill. John spoke to Mary at four o'clock.
S Did Bill speak to Mary at four o'clock?
L Unknown

[2] Bill MacCartney contributed a package in which 246 of the questions are converted to declarative hypotheses. These data are suitable for evaluating an entailment recognizer, as done in MacCartney and Manning (2007). See: `http://www-nlp.stanford.edu/~wcmac/downloads/`

In contrast to RTE, the focus in the FraCaS test suite is on inferences that can be explained by formal semantic theories, without recourse to either world knowledge or commonsensical reasoning. The test suite does not provide explanations specifically for inferences enabled by the data. The linguistic phenomena that trigger inferences are to be inferred from the test title-description, e.g., "Generalized Quantifiers / Monotonicity (upwards on first argument)".

2.5 Annotation Work

Garoufi (2007) proposed an annotation scheme entitled Annotating RTE (ARTE), for marking diverse linguistic phenomena that trigger inferences in the RTE datasets. Its aim is to achieve full coverage of entailment phenomena, thereby making it possible for human annotators to provide a complete analysis of sentences in the corpus. The scheme was applied to 500 pairs from RTE 2 and yielded insights into the inference-triggering phenomena it displays. The rationale behind the project is twofold: It is assumed that a better understanding of the phenomena in the corpus can (a) improve datasets in the future, and (b) allow an analysis of textual entailment recognizers with respect to different types of inferences, thereby augmenting textual-entailment technology. Garoufi also reports an evaluation of recognizers participated in the 2nd RTE challenge.

As illustrated in (7), the scheme analyzes positive pairs at three levels: *Alignment*, *Context*, and *Coreference*. Negative pairs are subjected to a more basic analysis.

(7) *t*: Since it$_1$ saw the light of day$_2$ in 2004$_3$, Katamari Damacy$_{1-coref}$ has gone on to become one of the biggest cult hits in the history of video games.

h: Katamari Damacy$_1$ was released$_2$ in 2004$_3$.

According to the scheme, the phrases that are underlined in t and h are connected at the alignment level. For example, *saw the light of day* in t is aligned to *was released* in h. At the coreference level, *Katamari Damacy* in t and *it* in t are marked as co-referring. Additionally, the scheme specifies different sub-types of alignment. Thus, cases in which the aligned phrases are very similar are marked as *identical*; alignment between allomorphic genitive phrases (possessive pronoun, the clitic 's and the preposition *of*) are marked as *genitive*. The specifications at the context and coreference levels are also sub-categorized.

Bentivogli et al. (2010a) proposed a methodology for creating specialized entailment data sets by documenting isolated linguistic phenomena that license entailment. Their study used entailment pairs from the RTE. For each of these, a set of "mono-thematic" pairs was generated containing only one specific phenomenon that contributes to the original entailment. As part of a feasibility study, this methodology was applied to a sample of 90 pairs randomly extracted from RTE 5. The phenomena were divided into broad classes (lexical, syntactic, etc.) and then into more fine-grained categories (synonymy, active-passive, etc.). For each of these categories, a general inference rule was defined, e.g., *argument realization*: "x's y" → "y of x". The rationale was to provide the corpus with a detailed analysis of the entailments, as opposed to the single annotation of *entailing* vs. *non-entailing* used hitherto; the goal was to help developers of entailment recognizers to improve the performance of their systems. However, a typology such as the one suggested by Bentivogli et al. (2010a) requires substantial human input to analyze the pairs and to verify the accuracy and consistency of the annotation.

Sammons et al. (2010) concur that improving the performance of recognizers would require annotations that label various inference-triggering phenomena. Yet, in their view, the aim should be to identify inferential phenomena predominant in the RTE a project that would involve a large-scale distributional effort. The results should then be shared with the RTE community, providing resources that would allow a more detailed assessment of RTE systems which would ultimately enhance their performance.

In keeping with this rationale and relying on insights derived from successful RTE systems, Sammons et al. proposed a model for identifying and annotating inference in entailment data. According to their conclusion, a pilot annotation study conducted to test the model confirmed the feasibility of their approach and attested to its contribution to the understanding of the challenges in the RTE task.

2.6 A Look Forward

This thesis builds on and further develops the rationale expounded in Garoufi (2007), Bentivogli et al. (2010a) and Sammons et al. (2010). Chapter 3 describes an annotation project of identifying and annotating inference-licensing phenomena that are present in roughly 80% of the positive examples in the RTE: appositive, restrictive and intersective types of modification. This annotation work is released as a corpus entitled Semantic Annotation of Textual Entailment (SemAnTE) 1.0, which is freely available on the web, adding to the other, above mentioned annotation projects.

Chapters 4–6 explore an alternative paradigm for investigating textual entailment, which operates with texts that are more restricted compared to the RTE corpora, but which employs a more powerful annotation methodology that enables the compilation of a theory-based corpus of entailment data. As the first step within this paradigm, we resort to a semantic model of entailment to account for the modification phenomena studied which allows us to explain inferential processes formally. Second, we implement the model in an annotation platform which concomitantly serves as a proof system. In a semi-automatic workflow, the platform allows human annotators to apply the semantic model to the entailment data at hand. Third, we use the platform to create a new theory-based corpus of entailment, in which all inferential processes addressed, or lack thereof, are explained formally by the semantic model employed.

CHAPTER 3

Predominant Modification Phenomena in the RTE

This chapter is based on Toledo et al. (2012) and parts of it also appeared in Toledo et al. (2013a) and Toledo et al. (2013b).

Overview

This chapter addresses the first research question of this thesis: whether and to what extent inferences stemming from modification phenomena are common in the RTE corpora. The focus is on restrictive modification and its intersective and appositive varieties. Informal syntactic definitions are formulated for these phenomena and used for identifying inferences in concrete examples. Annotation work that probes the frequency of these phenomena in RTE 1–4 is described next. It is demonstrated through a quantitative analysis of the findings that inferences triggered by modification are easy to identify and that they are found in the majority of positive pairs of RTE 1–4. Challenges encountered in the annotation work and its evaluation, as well as in systematically accounting for inferential processes, provide the motivation for the development of a more comprehensive annotation paradigm, presented in the next chapter.

3.1 Introduction

Recognizing an entailment relation in a *t*–*h* pair involves identifying a sequence of local inferences leading from the text to the hypothesis. This sequence constitutes an inferential process, or path, for the pair in question. Annotators with training in linguistics are able to isolate local inferences and to describe processes as part of their subjective interpretation of entailments. Findings indicate that many of the local inferences found in such processes in the RTE corpora can be explained by the semantics of modification phenomena, described in what follows.

3.2 Modification Phenomena

Structures that fall under the rubric of modification can be informally described as configurations where a syntactic constituent X combines with another such constituent Y yielding an expression $X\ Y$ of the same category as Y. In such configurations, X, which is termed a *modifier* of Y, can be modeled semantically as an operator OP. The role of the latter is to map Y's meaning, or *extension*, M to a meaning (extension) $OP(M)$ of the same semantic type as M. We call OP a *modifier function*. In what follows, we analyze the relation between $OP(M)$ and M.

Instances where the extension $OP(M)$ *subsumes* M are termed *restrictive modification*. Such a subsumption relation between expressions will be standardly modelled in Chapter 4 as a partial order between their denotations. Consider for example the word *boy* and the modification *tall boy* in *A tall boy jumps*. The meaning of the nominal expression *tall boy* semantically subsumes the meaning of the noun *boy*. Intuitively, this corresponds to the fact that the set of tall boys is *contained* in the set of boys.

Two major sub-classes of restrictive-modification phenomena are distinguished here:

- Direct Restrictive Modification: cases where the extension $OP(M)$ directly combines with the meanings of other constituents in the sentence. For example, *tall* in *Jan is a tall boy*, is interpreted directly as part of a predicative NP in the subject of the sentence. This is the most basic type of restrictive modification.

- Appositive Modification: cases where $OP(M)$ does not directly combine with the meanings of other constituents in the sentence. Rather, the information that $OP(M)$ conveys is added at the contextual level. For instance, consider the modification of *Barack Obama* by *the 44th president of the US* in (8).

  (8) Barack Obama, the 44th president of the US, has been a Senator from Illinois.

  This modification is interpreted as the proposition in (9), irrespectively of the predication *has been a Senator from Illinois*. This proposition is a presupposition of the original sentence.[1]

(9) Barack Obama is the 44th president of the US.

One prominent subclass of direct restrictive modification is when $OP(M)$ can be described as an intersection of M and some other extensions M' : $OP(M) = M \wedge M'$. In such cases the restrictive modifier can also be classified as *intersective.* For example, *Dutch* is an intersective modifier of *boy* in *Jan is a Dutch boy.* The modification *Dutch boy* can be described as an intersection of the meanings of *Dutch* and *boy*, and it follows that *Jan is a boy* and that *Jan is Dutch.* In restrictive modifications where $OP(M)$ is not an intersection of M and M', as in *Jan is a tall boy*, it only follows that *Jan is a boy* but not that *Jan is tall.* Jan may be a tall boy without being considered "tall".

In all the above types of modification, the meaning of the modification construction (*tall/Dutch boy*, *President Obama*) subsumes the meaning of the modified element (*boy, Obama*). All these types of modification are standardly referred to as *restrictive.* However, for the sake of brevity, Direct restrictive modifiers that are not intersective are considered here as the default case. Therefore, only these modifiers are referred to as "restrictive". The other two types of restrictive modification are referred to as "appositive" and "intersective". In as much as all the three types are restrictive, they give rise to common inferential processes, as will be demonstrated below.

Notably, some modifiers in natural language are non-restrictive. For example, the adverb *almost* in *The room is almost full* and the adjective *alleged* in *John is an alleged killer.* Such modifiers create meanings that do not subsume the meaning of the modified element. We address non-restrictive modifiers in Chapter 4. See Partee (1995) for a typology of modifiers that distinguishes between restrictive (subsective), intersective, non-restrictive (nonsubsective) and privative adjectives.

3.2.1 Restrictive Modification

The meaning of a restrictive modification construction X Y subsumes that of the modified element Y. The upshot is that texts incorporating a modification construction X Y often support hypotheses where the modified element Y appears without the modifier X. We call this type of inference *modifier-drop*, as exemplified in (10).[2]

(10) *t*: The controversy-racked oil giant Shell has named a new head of finance in an effort to calm nervous shareholders.
h: Shell is an oil giant.

[1] The reasoning is that it is left untouched under (a) sentential negation or modal insertion: *Barack Obama, the 44th president of the US has not/might have been a Senator*, (b) a reformulation as a question: *Has Barack Obama, the 44th president of the US, been a Senator?*

[2] All the examples in this chapter are taken from RTE 1–4 except for (13), (14) and (22), which were composed for illustrating specific points.

In this example, *controversy-racked* acts as a restrictive modifier of *oil giant.* A standard analysis of this pair posits a local inference from *controversy-racked oil giant* in t to *oil giant* in h.

The examples below present additional syntactic configurations in which an inferential process stems from restrictive modification: a prepositional phrase modifying in (11) a verb phrase and in (12) a noun phrase.

(11) *t*: The watchdog International Atomic Energy Agency meets in Vienna on September 19.
h: The International Atomic Energy Agency holds a meeting in Vienna.

(12) *t*: U.S. officials have been warning for weeks of possible terror attacks against U.S. interests.
h: The United States has warned a number of times of possible terrorist attacks.

It should be noted at this point that, in many examples, the modifier X and the modified element Y may take a different form in the entailment from t to h due to a lexical, morphological or syntactic inference. For instance, in (11), Y in t, *meets in Vienna*, becomes *holds a meeting in Vienna* in h. Similarly, *possible terror attacks* in (12) becomes *possible terrorist attacks.* We analyze these examples as instances of restrictive modification on the assumption that, in each case, the inference stemming from modification and the inference stemming from lexical knowledge are independent parts of the full inferential process from t to h. This chapter does not analyze inferences unrelated to modification phenomena, but the relevant lexical and other inference-licensing constructions are presented in Chapter 4 as part of the semantic model proposed.

Importantly, an inference derived from a drop of a constituent does not, in itself, indicate that the modification is restrictive: It is neither a necessary nor a sufficient condition for restrictivity. Thus, (13) shows a constituent-drop inference between *a student or a teacher*, on the one hand, and *a student*, on the other, although *a student or a teacher* is not a restrictive modification.[3]

(13) *t*: Every person who is a student or a teacher smiled.
h: Every person who is a student smiled.

Furthermore, not all instances of restrictive modification license a modifier-drop inference. For example, the restrictive modification in (14) does not license entailment.

(14) *t*: Every tall student smiled.
h: Every student smiled.

[3] The coordinating structure *a student or a teacher* is not subsumed by the phrase *a student.* Note that: *John is a student or a teacher* does not entail that *John is a student.*

Our analysis at this stage is informal, as its purpose is to explain modification intuitively and to demonstrate a range of simple inferences thus perceived. A rigorous formal semantic analysis to account for all the cases addressed in this chapter will be developed in Chapter 4.[4]

3.2.2 Intersective Modification

The meaning of an intersective-modification construction $X\ Y$ is the same as the intersection of the extensions of Y and X. Accordingly, texts that include an intersective-modification construction $X\ Y$ often support hypotheses containing only one of the elements, as in (15):

(15) *t*: The Zulu are an African ethnic group of about 11 million people who live mainly in KwaZulu-Natal Province, South Africa.
h: The Zulus live in Kwazulu-Natal Province.

In this pair, the inferential process includes a local inference from the intersective modification in the underlined relative clause in t to the underlined verb phrase in h. The modified element is *an African ethnic group of about 11 million people.* The modifier is the relative clause *who live mainly in KwaZulu-Natal Province, South Africa.*

Another common syntactic construction that lends itself to an intersective analysis is predicate conjunction, as in (16):

(16) *t*: Nixon was impeached and became the first president ever to resign on August 9th 1974.
h: Nixon was the first president ever to resign.

In this example we infer *was the first president ever to resign* in h from *was impeached and became the first president ever to resign* in t.

3.2.3 Appositive Modification

In many entailments, appositive modification in a construction $X\ Y$ licenses an inference that attributes a property associated with the modifier X to an entity associated with Y, as in (17).

(17) *t*: The incident in Mogadishu, the Somali capital, came as U.S. forces began the final phase of their promised March 31 pullout.
h: The capital of Somalia is Mogadishu.

In this entailment, the constituent *Mogadishu, the Somali capital* in t gives rise to the conclusion that the Somali capital is Mogadishu. As in this example, appositive modifiers are typically separated by a pair of commas or dashes. A similar pattern is manifested in (18).

[4] The phenomena in (13) and (14) are direct results of the downward monotonicity of *every* on its first argument (*person* and *student*, respectively), which is a property of universal quantification as treated in the standard semantic theory.

(18) *t*: A senior coalition official in Iraq said the body, which was found by U.S. military police west of Baghdad, appeared to have been thrown from a vehicle.

h: A body has been found by U. S. military police.

Appositive modification often obtains in phrases denoting titles, as illustrated in (19).

(19) *t*: Prime Minister Silvio Berlusconi was elected March 28 with a mandate to reform Italy's business regulations and pull the economy out of recession.

h: The Prime Minister is Silvio Berlusconi.

Unlike the previous two examples, here the appositive modifier (*prime minister*) represents a title and is therefore not punctuationally separated from the modified entity (*Silvio Berlusconi*).

As a sub-class of restrictive modification, appositives are subject to a modifier-drop inference. For example, in (20), the modifier *Iamgold's chief executive officer* is dropped, and in (21), the modifier *The country's largest private employer* is dropped.

(20) *t*: Mr. Conway, Iamgold's chief executive officer, said the vote would be close.

h: Mr. Conway said the vote would be close.

(21) *t*: The country's largest private employer, Wal-Mart Stores Inc., is being sued by a number of its female employees who claim they were kept out of jobs in management because they are women.

h: Wal-Mart sued for sexual discrimination.

Appositive modification can also account for a combination of inferences, as illustrated in (22).

(22) *t*: Leonard Cohen, the famous singer-songwriter, celebrated his 80th Birthday with a new album.

h: A famous singer-songwriter released a new album

Our analysis of the inferential path in this example relies on two inferences: (a) a modifier drop, which leads to the inference *Leonard Cohen, the famous singer-songwriter* ⇒ *Leonard Cohen*, and (b) attributing to the entity the information conveyed by the modifier, thus Leonard Cohen *is* the famous singer-songwriter. Consequently, applying (a) results in *Leonard Cohen celebrated his 80th Birthday with a new album*; and applying (b) results in *The famous singer-songwriter celebrated his 80th Birthday with a new album.* We can then infer that *A famous singer-songwriter celebrated his 80th Birthday with a new album* based on the semantics of the definite article, which we analyze in Chapter 4. The rest of the inference relies on lexical knowledge.

Interaction between different modification phenomena

Inferential processes in RTE examples are often based on a combination of two or more of the modification phenomena described above. The example in (23) shows an inference that combines all three types of restrictive-modification

(23) *t*: The anti-terrorist court found two men guilty of murdering Shapour Bakhtiar and his secretary Sorush Katibeh, who were found with their throats cut in August 1991.

h: Shapour Bakhtiar died in 1991.

We can construct the following inferential path by appealing to the semantics of modification phenomena:

- First, an inference from the relative clause *Shapour Bakhtiar and his secretary Sorush Katibeh, who were found with their throats cut in August 1991* to *Shapour Bakhtiar and his secretary Sorush Katibeh were found with their throats cut in August 1991* (appositive modification).

- Next, an inference from *August 1991* to *1991* (restrictive modification of *1991* by *August*).

- Finally, an inference from *Shapour Bakhtiar and his secretary Sorush Katibeh* to *Shapour Bakhtiar* (intersective modification).

By combining these three local inferences, it is possible to conclude that *Shapour Bakhtiar was found with his throat cut in 1991.* Additional world knowledge is required to infer that *found with his throat cut* entails *died* - whereupon the entailment can be fully validated.

3.3 Prevalence of Modification

3.3.1 Van Strien (2009)

The prevalence of appositive and restrictive modification in the RTE datasets was first noted by Van Strien (2009). In this work, instances of appositive and restrictive modification were annotated in positive pairs from the development sets of RTE 1-3 where at least one local inference stemming from either type of modification led a human annotator to recognize entailment. For example:

(24) *t*: To the world, M. Larry Lawrence, the new U.S. emissary to Switzerland who hosted President Clinton on his Southern California vacation, will be known as Mr. Ambassador.

h: Larry Lawrence is the head of the U.S. Embassy in Switzerland.

(25) *t*: Two persons were injured in dynamite attacks perpetrated this evening against two bank branches in this northwestern, Colombian city.

h: Two bank branches were attacked with dynamite.

In (24), the human annotator marked the underlined appositive construction as it licenses the identification of *M. Larry Lawrence* as *the new U.S. emissary to Switzerland*, which is required for deriving h from t. In (25), the annotator marked the underlined restrictive modification, recognizing that *in this northwestern, Colombian city* restrictively modifies *two bank branches* and is thus required for inferring h from t.

Quantitative analysis of the annotations indicates that, in 49.2% of the pairs (538 out of 1094), the annotator interpreted the entailments as including a restrictive modification inference, and in 24.5% of the pairs (268 out of 1094) - an appositive modification inference. In total, in 64.4% of the pairs at least one of these phenomena led the annotator to recognize entailment. These figures indicate that inferences stemming from appositive and restrictive modification are prevalent in the development and test sets of RTE 1–3.

3.3.2 SemAnTE 1.0

Semantic Annotation of Textual Entailment (SemAnTE) 1.0 is a pilot annotation project that addresses appositive, restrictive and intersective modification in the positive pairs of the development and test sets of RTE 1-4. The aim was to substantiate Van Strien's (2009) findings and, in addition, to target intersective modification as well. The results constitute the empirical basis for developing a more comprehensive and precise semantic model. The annotation work was carried out by two Master's students of Linguistics.

Annotation Guidelines

Annotators were instructed to mark only those modification phenomena that were operational in an inferential process between the text and the hypothesis. The annotation guidelines were formulated as follows. For every positive pair from RTE 1–4,

1. Read the data and verify the entailment intuitively.
2. Describe informally why the entailment holds.
3. Annotate all the instances of restrictive, intersective and appositive modification that contribute to the inferential process.

For each modification phenomenon, annotators marked its type and aligned its expression in the text and in the hypothesis: see underlined parts of the text in Examples (10)–(12) and (15)–(21).

	Dev set			Test set		
Annotation	A$_{\#}$	P$_{\#}$	P$_{\%}$	A$_{\#}$	P$_{\#}$	P$_{\%}$
Appositive	97	87	31	161	134	34
Restrictive	180	124	44	243	167	42
Intersective	90	79	28	126	112	28
Any	367	210	74	530	297	74

(a) RTE 1

	Dev set			Test set		
Annotation	A$_{\#}$	P$_{\#}$	P$_{\%}$	A$_{\#}$	P$_{\#}$	P$_{\%}$
Appositive	179	149	37	155	135	34
Restrictive	314	205	51	394	236	59
Intersective	141	119	30	161	144	36
Any	634	318	80	710	350	88

(b) RTE 2

	Dev set			Test set			Test set		
Annotation	A$_{\#}$	P$_{\#}$	P$_{\%}$	A$_{\#}$	P$_{\#}$	P$_{\%}$	A$_{\#}$	P$_{\#}$	P$_{\%}$
Appositive	188	150	38	166	136	34	259	200	40
Restrictive	300	201	50	307	193	48	429	271	54
Intersective	176	138	35	162	134	34	192	164	33
Any	664	329	82	635	328	82	880	413	83

(c) RTE 3 (d) RTE 4

Table 3.1: Figures of Annotations in SemAnTE 1.0

Annotation Results

In RTE 1-4, modification constructions were annotated in 80.65% of the positive entailment pairs. The full data appear in Table 3.1, separated according to the development and test sets of each corpus. The rubric $A_{\#}$ indicates the number of annotations, $P_{\#}$ - the number of entailment pairs that were annotated, and $P_{\%}$ - the percentage of annotated pairs relative to the total amount of entailment pairs.

These results confirm that appositive, restrictive and intersective types of modification are prevalent entailment-licensing phenomena in the RTE corpora. In Appendix C we describe a use case of SemAnTE 1.0 in which the corpus serves as a gold standard of modification. Using the corpus we evaluate the application of rules that derive inferences from these phenomena in inferential processes generated by the BIUTEE system (Stern and Dagan, 2011), described previously in Chapter 2.

Consistency Checks

Four informal cross-annotator consistency checks were performed. Each informal check examined 50-70 pairs addressed by both annotators independently. These informal checks are reported in Table 3.2.

We did not use the Kappa score since it is not informative for this type of annotation and for the analysis we were interested in. First, the annotation task requires more than a classification among a definite set of options: annotators also mark the boundaries of the manifestation of each phenomenon in the text and hypothesis, and specify its internal structure (e.g. what is the modifier and what is the modified element, in the case of restrictive modification). The checks examine all of these parameters and consider two annotations as identical only if all parameters match.

An alternative approach is to create a gold standard of annotations and then to compare the annotations marked by each annotator to the gold standard. This analysis yields precision and recall scores for each annotator. However, as soon as we started to perform consistency checks we learned that in many examples there is a number of valid ways to analyze a pair. Due to that it is difficult to talk about a single "correct" annotation in every given case.

For these reasons we opted for open-book checks, in which we judge the annotations that each annotator marked in the context of the inferential path that she assumed in her subjective interpretation of the data. We compare the identical annotations and also analyze the divergent and erroneous ones.

The results of these checks show that about 68% of the annotations (297 out of 437) are identical. About 25% of the annotations (109 our of 437) differ due to (a) more than one option to understand a sentence structure (*Ambig.-Struct.*), (b) several possibilities to analyze an inference (*Ambig.-Infer.*), and (c) or insufficiently specific annotation scheme (*Ambig.-Scheme*). Importantly, annotations diverging on account of such types of ambiguity are legitimate and do not count as errors. The measures are reported in Table 3.2. Only about 9.5% of the annotations (41 out of 437) were judged incorrect by at least one of the annotators, who questioned either the phenomenon marker or the specification of the scope of the phenomenon marked (*Incorrect Ann.*).

To elucidate reasons for diverging annotations, let us examine several representative examples involving the three different types of ambiguity listed above.

- *Ambig.-Struct.* – annotations diverging due to structural ambiguity.

(26) *t*: The British government has indicated its readiness to allow Argentine companies to take part in the development of oilfields in the Falkland islands' territorial waters.

h: The British government is ready to allow Argentine companies to participate in the development of oilfields.

Annotator 1: Marked *in the Falkland islands' territorial waters* as a modifier of *development of oilfields*, corresponding to the structure: [[develop-

Measure	RTE 2	RTE 1+2	RTE 3	RTE 4
Data Source(s)	Dev set	Test sets	Dev+Test sets	Test set
Entailment Pairs	50	70	70	70
Total Ann.	93	112	99	133
Identical Ann.	62	83	66	86
Missing Ann.	2	7	7	10
Incorrect Ann.	10	1	2	2
Ambig.-Struct.	9	16	20	15
Ambig.-Infer.	N/A	8	13	12
Ambig.-Scheme	N/A	0	9	7
Consistency (%)	66.67	74.11	66.67	64.67

Table 3.2: Informal Consistency Checks in SemAnTE 1.0

ment of oilfields] [in the Falkland islands' territorial waters]].

Annotator 2: Marked *in the Falkland islands' territorial waters* as a modifier of *oilfields*, corresponding to the structure: [development of [oilfields [in the Falkland islands' territorial waters]]].

- *Ambig.-Infer.* – annotations diverging due to multiple paths for deriving the inference.

(27) *t*: Microsoft Corp., on Thursday, posted higher quarterly earnings as revenue rose 12 percent, but its shares fell after the world's largest software market said current quarter sales would fall below Wall Street expectations.

h: Microsoft showed revenue growth.

Annotator 1: Inferred *showed revenue growth* from *posted higher quarterly earnings* and therefore marked *as revenue rose 12 percent* as a restrictive modifier of *posted higher quarterly earnings.*

Annotator 2: Inferred *showed revenue growth* from *posted higher quarterly earnings, as revenue rose 12 percent* and therefore did not mark a restrictive modifier in this construction.

- *Ambig.-Scheme.* – annotations diverging due to insufficiently specific annotation scheme.

(28) *t*: Clonaid said, Sunday, that the cloned baby, allegedly born to an American woman, and her family were going to return to the United States Monday, but where they live and further details were not released.

h: Clonaid announced that mother and daughter would be returning to the US on Monday.

> Problem Description: The annotation guidelines do not specify how to mark modification of a non-continuous element. In this case, *Sunday* modifies the combination of *said* and its complement *that the cloned baby, allegedly born to an American woman* but this annotation cannot be marked because the modified element comprises two separate syntactic constituents.
>
> Annotator 1: Annotated only *said* as the modified element in a restrictive modification annotation.
>
> Annotator 2: Did not mark the modification.

In sum, about 93% of the annotations that were checked can be considered as correct. This informal check suggests that the modification phenomena addressed in this study can be reliably annotated.

Corpus Release

SemAnTE 1.0 is freely available on the Web.[5] The format of the resource is described in Appendix B. It adds to existing annotation projects carried out on the RTE datasets, such as Garoufi (2007) and Bentivogli et al. (2010a), mentioned in Chapter 2.

3.4 Discussion

The work on SemAnTE 1.0 led to two important conclusions that influenced the course of our research. First, it provided an answer to this thesis's first research question, in as much as restrictive modification and its intersective and appositive varieties were found to be prevalent in the RTE corpora. This finding, in turn, led us to assume that modification phenomena are sufficiently common in natural language entailment to serve as a focus of further analysis of textual entailment in our investigation.

Second, the process of annotating inference-enabling phenomena and verifying the accuracy of the annotations marked proved to be challenging and time consuming. This was to be expected: the workflow requires annotators to create an abstract inferential path that leads from the text to the hypothesis and then to mark the modification structures that occur in this path. The analysis is informal and involves translating an intuitive entailment judgment to a sequence of concrete linguistic phenomena. An annotator may therefore miss one or more such structures or annotate it inaccurately, in particular when an entailment involves an intricate combination of inference triggers. No less importantly, the accuracy of annotations has to be examined manually, in light of an annotator's subjective interpretation of each entailment. Moreover, the high chance of obtaining divergent (but correct) annotations rendered inter-annotator agreement (IAA) checks less than optimal for indicating the annotation reliability

[5] See: `http://logiccommonsense.wp.hum.uu.nl/resources`

and for detecting annotation errors. Thus the task proved to be labor intensive and subject to human error.

From the problems encountered in using an informal annotation scheme, it was concluded that analyzing entailment phenomena requires a formal model. As a basic framework for such a model, we adopted state-of-the-art formal semantics. This leads to the main research question of this thesis: how can formal semantic theory be implemented in an annotation platform that would facilitate the modeling of textual entailment. This question is investigated in Chapter 4, which introduces a formal model of entailment, and in Chapter 5, which describes its implementation in an annotation platform.

CHAPTER 4

Formal Model of Entailment and Annotation

This chapter is a development of Toledo et al. (2013b) and parts of it also appeared in Toledo et al. (2013a).

Overview

This chapter describes a formal model of entailment that uses simple annotations of standard syntactic trees. These annotations are marked using lexical items from a pre-defined lexicon and establish a connection between parse trees and semantic representations with a model-theoretical interpretation. The deductive power of this approach is illustrated with RTE examples by binding linguistic phenomena to semantic representations and then providing a theoretical analysis that explains how the entailment follows. This prepares the ground for the introduction of the proof-based annotation system in Chapter 5.

4.1 Theoretical Context

The natural-language entailment model presented here is premised on the assumption that entailment describes a *preorder* relation on natural-language sentences. Thus, any sentence trivially entails itself (reflexivity), and given two entailments $T_1 \Rightarrow H_1$ and $T_2 \Rightarrow H_2$ where H_1 and T_2 are identical sentences, $T_1 \Rightarrow H_2$ (transitivity).

A computational theory of entailment should describe an approximation of this preorder. To this end, a standard model-theoretic extensional semantics is adopted, based on the simple *partial order* on the domain of *truth-values*. Accordingly, each model M assigns sentences a truth-value in the set $\{0, 1\}$. Tarskian semantics is adopted, whereby a theory of entailment is considered adequate if the intuitive entailment preorder on sentences in natural language is described in the model as the pairs of sentences T and H whose truth-values $[\![T]\!]^M$ and $[\![H]\!]^M$ satisfy $[\![T]\!]^M \leq [\![H]\!]^M$ for all models M. The theory proposed here is developed to meet this condition.

4.2 Semantic Essentials

The theoretical model is established using a standard semantic framework, as in Dowty et al. (1981), Gamut (1991) and Winter (2010), where the basic linguistic abstraction is the concept of *type*. Types are assigned to natural language expressions and encode general aspects of their meaning.

Definition 1 Let B be a finite non-empty set of basic types. The set of $\mathsf{functional\ types}$ over B is the smallest set $\mathcal{T}^{\mathsf{B}}$ that satisfies:

(i) $\mathsf{B} \subseteq \mathcal{T}^{\mathsf{B}}$;

(ii) If τ and σ are types in $\mathcal{T}^{\mathsf{B}}$ then $(\tau \to \sigma)$ is also a type in $\mathcal{T}^{\mathsf{B}}$.

The '$\to$' symbol is omitted for perspicuity, and parentheses in types are erased whenever ambiguity does not arise. Thus, the functional type $(\tau \to \sigma)$ is denoted '$\tau\sigma$'. For our purposes we use $\mathsf{B} = \{e, t\}$, where e is the type of *entities* and t is the type of *truth-values*.

Table 4.1 contains representative examples of types that are commonly used for various natural-language expressions. These types are illustrated in the interpreted lexicon described in Section 4.3. The type scheme $\tau\tau$ for modifiers and the type scheme $\tau(\tau\tau)$ for coordinators are used for different expressions, with different types instantiating $\tau \in \mathcal{T}^{\{e,t\}}$.

For each type τ in $\mathcal{T}^{\mathsf{B}}$ a corresponding *domain* D_τ is inductively defined. For basic types in B the corresponding domains are assigned by assumption. For each non-basic functional type $\tau\sigma \in \mathcal{T}^{\mathsf{B}}$ the corresponding domain is defined by:

$$D_{\tau\sigma} \;=\; D_\sigma^{D_\tau} \;=\; \text{the set of functions from } D_\tau \text{ to } D_\sigma$$

The domain D_t of truth-values is assumed to be constant: the set $\{0, 1\}$ with the natural partial order $\leq$. This partial order allows us to capture semantically the preorder relation induced by entailment on natural language sentences. Given a non-empty domain of entities $D_e = E$, the collection of domains $\mathcal{F}^E = \{D_\tau | \tau \in \mathcal{T}^{\{e,t\}}\}$ is termed the *frame over* E. Thus, the frame over E contains the respective domain E of entities, the domain of truth-values, and all the one-place functions that are derived from these two sets. Table 4.2 lists examples for domains in $\mathcal{F}^E$with the type of their members. $\mathbf{2} = \{0, 1\}$ is abbreviated.

Type	Expression
t	sentences
e	proper names, referential noun phrases
et	intransitive verbs and common nouns
$e(et)$	transitive verbs
$(et)t$	quantificational noun phrases
$(et)e$	definite article (*the*)
$(et)((et)t)$	determiners (*some, every*)
$\tau\tau$	modifiers (adjectives, adverbs, prepositional phrases, negation, relative clauses)
$\tau(\tau\tau)$	coordinators (conjunction, disjunction, restrictive relative pronouns)

Table 4.1: Types Commonly Used for Natural Language Expressions

Domain	Type
E	e
E^E	ee
$\mathbf{2}$	t
$\mathbf{2}^\mathbf{2}$	tt
$E^\mathbf{2}$	te
$\mathbf{2}^E$	et
$\mathbf{2}^{(E^E)}$	$(ee)t$
$(\mathbf{2}^E)^E$	eet
$E^{(\mathbf{2}^\mathbf{2})}$	$(tt)e$
$(E^\mathbf{2})^\mathbf{2}$	tte

Table 4.2: Domains in $\mathcal{F}^E$with the Types of their Members

Let a *lexicon* Σ be a set of terminal symbols (=words). A *typing function* TYP is used for assigning a type to each word in Σ. Using this function and a set of entities E, a *model* M is defined over Σ. This is done by mapping each word in Σ to a denotation of the appropriate type in the frame $\mathcal{F}^E$. This mapping, which is termed an interpretation function, is defined in what follows. While types encode general aspects of an expression's meaning, denotations encode more specific aspects within its typed domain.

Definition 2 An $\mathsf{interpretation\ function}$ I over a lexicon Σ with a typing function TYP and a set E, is a function from the words in Σ to the set $\bigcup \mathcal{F}^E$, which sends every word $w \in \Sigma$ to an element in the corresponding typed domain: $I(w) \in D_{\mathsf{TYP}(w)}$.

For instance: let *see* be a word of type $e(et)$ in a typed lexicon Σ. Then for any non-empty set E, an interpretation function over Σ and E sends the word *see* to a function in $(\mathbf{2}^E)^E$: a function f from entities to functions from entities to truth-values. By standard currying (Curry and Feys, 1958), this function f characterizes a binary relation over E. Informally speaking, this relation describes who sees who in a given model M.

A model is a pair consisting of a set of entities E and an interpretation function I, as formally defined below.

Definition 3 A model over a lexicon Σ with a typing function TYP is a pair $M = \langle E, I \rangle$ where E is a non-empty set and I is an interpretation function over the typed lexicon Σ and the set E.

Models over a lexicon are used to assign denotations to complex expressions over this lexicon. Given a language L over Σ and a model $M = \langle E, I \rangle$ over Σ, every parsed expression *exp* in L is assigned a type and a *denotation* $[\![exp]\!]^M$. Every terminal expression w in Σ is assigned the type $\mathsf{TYP}(w)$ and the denotation $[\![w]\!]^M = I(w)$. For non-terminal expressions, a binary parse is assumed. Thus, any non-terminal expression *exp* in L is assumed to be parsed into two sub-expressions exp_1 and exp_2 (not necessarily in this order). Such complex expressions are assigned a type and a denotation in M subject to the restriction that The types of exp_1 and exp_2 are $\tau\sigma$ and τ, respectively. With this restriction, types and denotations of complex expressions are defined through function application. Thus, the type of the expression *exp* is σ, and its denotation $[\![exp]\!]^M$ in M is the following element in D_σ:

$$[\![exp]\!]^M = [\![exp_1]\!]^M([\![exp_2]\!]^M), \text{ where } [\![exp_1]\!]^M \in D_{\tau\sigma} \text{ and } [\![exp_2]\!]^M \in D_\tau$$

In order to describe lexical denotations of words, interpretation functions within models are restricted by *ad hoc* assumptions, which are standardly expressed in higher order lambda-calculus (see Section 4.3). Models that satisfy these *ad hoc* restrictions on lexical denotations are referred to as *intended models.*

A lexicon Σ, together with a typing function TYP and a specification of intended models, is referred to as an *interpreted lexicon.* Assume now that T and H are parsed text and hypothesis sentences of type t over an interpreted lexicon Σ. *Logical entailment* between T and H is model-theoretically described below as the classical Tarskian property.

The truth-conditionality criterion: *Let T and H be parsed expressions of type t, over an interpreted lexicon Σ. The parsed sentence T* logically entails *H if and only if the relation $[\![\mathrm{T}]\!]^M \leq [\![\mathrm{H}]\!]^M$ holds between the truth-value denotations of T and H in all intended models M.*

Simple Example

Consider a mini-lexicon $\Sigma = \{$*Dan, sat, ate, and*$\}$ with the types e, et, et and $(et)(et)et$ respectively. Assume intended models where the interpretation function I assigns the words *Dan, sat* and *ate* arbitrary denotations **dan**, **sit** and **eat**, and where the word *and* is assigned the denotation given below in any model. We use the notation $\wedge_{ttt}$ to denote logical conjunction over the domain D_t of truth-values.
$I(and) = \textsc{and} = \lambda A_{et}.\lambda B_{et}.\lambda x_e.B(x) \wedge A(x)$

Given the natural binary parses, these intended models explain the entailment *Dan sat and ate* ⇒ *Dan ate*, since for each intended model M:

$$\begin{aligned}
&[\![\, \textit{Dan [sat [and ate]]} \,]\!]^M && \\
&= ((\textsc{and}(\mathbf{eat}))(\mathbf{sit}))(\mathbf{dan}) && \text{analysis} \\
&= (((\lambda A_{et}.\lambda B_{et}.\lambda x_e.B(x) \wedge A(x))(\mathbf{eat}))(\mathbf{sit}))(\mathbf{dan}) && \text{definition of } \textsc{and} \\
&= \mathbf{sit}(\mathbf{dan}) \wedge \mathbf{eat}(\mathbf{dan}) && \text{function app.} \\
&\leq \mathbf{eat}(\mathbf{dan}) && \text{definition of } \wedge \\
&= [\![\, \textit{Dan ate} \,]\!]^M && \text{analysis}
\end{aligned}$$

4.3 An Interpreted Lexicon

The lexicon contains a list of pre-defined lexical items that label the linguistic terms that are modeled semantically. These items cover both syntactic terms such as V_1 for intransitive verbs and V_2 for transitive ones, and semantic terms such as MR for restrictive modification and MI for intersective. Denotations for these lexical items are assigned by the interpretation function I.[1]

The lexicon is used for marking linguistic phenomena in the data – by binding words and phrases to corresponding lexical items. The items and their denotations are explained in Sections 4.3.3 and 4.3.4 below. Several basic functions and quantifiers, as well as the technique of entity lifting, are presented first.

4.3.1 Basic Functions and Quantifiers

The denotations in the lexicon rely on several basic functions and quantifiers:

- Logical conjunction, disjunction and implication over the domain D_t of truth-values are indicated by $\wedge_{ttt}$, $\vee_{ttt}$ and $\rightarrow_{ttt}$ respectively.

- Entity equality is indicated by $=_{eet}$. This function returns the truth-value 1 if the given entities are the same and 0 otherwise.

- Existential and universal quantification functions of type $(et)t$ are defined using the standard existential and universal quantifiers respectively:
  $\exists_{(et)t} = \lambda A_{et}.\exists x \in E.A(x)$
  $\forall_{(et)t} = \lambda A_{et}.\forall x \in E.A(x)$

- The $\iota_{(et)e}$ operator is defined for picking an entity from the characteristic set of a given predicate. The operator presupposes that the set is a singleton. For example, applying ι to **student** returns an entity from the set of students provided that the set contains that entity alone. The workings of

[1] The lexicon was designed in collaboration with Pepijn Kokke, and its implementation is in Kokke (2013b).

this operator within the treatment of entailment are explained in Section 4.3.5.

- The $\zeta_{(et)e}$ operator is defined for designating an entity that is a representative of the characteristic set of a given predicate. A representative of a set is not a member of a set; it is merely an entity that takes the set's part in the semantic analysis. For example, applying ζ to **fairytales** returns an entity that is a representative of the set of fairy tales in the analysis. This operator is explained within the treatment of entailment in Section 4.3.6.

4.3.2 Entity Lifting

Entity Lifting is a common technique for interpreting entities as generalized quantifiers – functions of type $(et)t$ – rather than bare arbitrary constants of type e. For instance, consider the word *John*. Its non-lifted interpretation is an arbitrary constant **john** of type e, while its lifted interpretation is the function $\lambda A_{et}.A(\mathbf{john}_e)$ of type $(et)t$ that returns 1 if and only if the set characterized by its argument contains **john**.[2]

Lifting switches the typical subject-predicate function application order: instead of a predicate taking an entity of type e as an argument, the lifted entity is a generalized quantifier taking the predicate as an argument. Note that the truth-values of these two application variants are the same since for any entity x of type e and for any predicate P of type et, applying P to x returns the same truth-value as applying $\lambda A_{et}.A(x)$ to P: $(\lambda A_{et}.A(x))(P) = P(x)$.

4.3.3 Constant Denotations

Constant denotations are non-arbitrary functions that are designed for modeling functional (grammatical) words in sentences. Table 4.3 lists the lexical items of these denotations with their type. Explanations are given below.

- The coordinator *and*, when instantiating a predicate conjunction as in the simple example *Dan sat and ate*, is analyzed as a function AND mapping any two et predicates A (*sat*) and B (*ate*) to a predicate that sends every entity x to the truth-value of the conjunction $A(x) \wedge B(x)$. This expresses intersective modification since the extension of the resulting predicate (*sat and ate*) is the intersection of the extensions of A and B. The coordinator may also appear in a plural construction such as the compound subject in *John and Mary sat* which has a distributive interpretation: *John sat* and *Mary sat*. For these cases *and* is analyzed as a function that maps

[2] We may use the terms "characteristic set" and "extension" interchangeably. Thus $\lambda A_{et}.A(\mathbf{john}_e)$ can also be described as a function that returns 1 if and only if **john** is in the *extension* of its argument A.

Label	Type	Denotation Function	Example	Remark
AND	$(et)ett$	$\lambda A_{et}.\lambda B_{et}.\lambda x_e.B(x) \wedge A(x)$	and	predicate conjunction
	$((et)t)((et)t)$ $(et)t$	$\lambda A_{(et)t}.\lambda B_{(et)t}.\lambda C_{et}.$ $A(C) \wedge B(C)$		distributive plurals conjunction
IS, V_AUX	$(et)et$	$\lambda A_{et}.A$	is, have	copula ‘be’ or auxiliary
IS$_{eq}$	$((et)t)((et)t)t$	$\lambda A_{(et)t}.\lambda B_{(et)t}.$ $A(\lambda x_e.B(\lambda y_e. = (x)(y)))$	is	equality ‘be’
IS$_{mod}$	$((et)et)et$	$\lambda A_{(et)et}.\lambda x_e.A(\lambda y_e.1)(x)$	is	modifier to predicate conversion
A, SOME	$(et)(et)et$	$\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.A(x)$ $\wedge B(x))$	a, some	existential quantifier
EVERY	$(et)(et)et$	$\lambda A_{et}.\lambda B_{et}.\forall x_e.A(x) \rightarrow B(x)$	every	universal quantifier
THE	$(et)(et)t$	$\lambda A_{et}.\lambda B_{et}.B(\iota(A))$	the	definite article
EMPTYDET	$(et)(et)t$	$\lambda A_{et}.\lambda B_{et}.B(\zeta(A))$		empty determiner
OF	$((et)t)$ $(((et)t)et)et$	$\lambda A_{(et)t}.\lambda B_{((et)t)et}.\lambda x_e.$ $B(A)(x)$	of	‘of’ preposition
POSS	$((et)t)(eet)e$	$\lambda A_{(et)t}.\lambda B_{eet}.$ $\iota(\lambda x_e.A(\lambda y_e.B(y)(x)))$	’s	subject possessive ’s
	$((et)t)(eet)$ $(et)t$	$\lambda A_{(et)t}.\lambda B_{eet}.\lambda C_{et}.$ $C(\iota(\lambda z_e.A(\lambda x_e.B(x)(z))))$		object possessive ’s
P_GR	$((et)et)(et)et$	$\lambda M_{(et)et}.M$	by	gerund preposition
WHO_R	$(et)(et)et$	$\lambda A_{et}.\lambda B_{et}.\lambda x_e.B(x) \wedge A(x)$	who	restrictive relative pronoun
WHO_A	$(et)(et)t$ $(et)t)$	$\lambda A_{et}.\lambda B_{(et)t}.\lambda C_{et}.C(\iota$ $(\lambda x_e.B(=(x)) \wedge A(x)))$	who	appositive relative pronoun
	$((et)t)(et)t$ $(et)t)$	$\lambda A_{(et)t}.\lambda B_{(et)t}.\lambda C_{et}.C(\iota$ $(\lambda x_e.B(=(x)) \wedge A(=(x))))$		appositive relative pronoun

Table 4.3: Pre-defined Denotations in the Interpreted Lexicon

two $(et)t$ generalized quantifiers, A and B, which commonly stand for two lifted entities, and a predicate C of type et, to the conjunction $A(C) \wedge B(C)$. When A and B are lifted entities, this practically means that the conjunction returns 1 if and only if the entities lifted by A and B make up a set that is a subset of the set characterized by C. Collective interpretations of plurals, even in the relatively simple sentence *John and Mary met in a dating site*, and certainly in the more complicated *Dan's girlfriend and Bill's boyfriend are Mary and John*, require a more complex analysis of the relation between the sets characterized by the entities and the predicates. These interpretations are not treated here.

- The copula *be*, including its inflections for tense and person (for example, *is* and *were*) is handled based on the type of the main predicate in the sentence. In sentences where the predicate is of type et, as in *Jan is reading* (see V_I below), the copula denotes the identity function on et predicates. This $(et)(et)$ function, which is labeled IS, maps each et predicate to itself.[3]
The copula can also express an equality relation between generalized quantifiers. For example, in sentences such as *Dan is Jan*, where the generalized quantifiers are lifted entities. The copula is analyzed as the equality relation $\textsc{is}_{eq}$, of type $((et)t)((et)t)t$.
In sentences with an adjectival predicate, such as *Dan is short*, where the predicate is of type $(et)et$, the copula is treated as a function of type $((et)et)et$ that converts a modifier into an et predicate.[4]
Auxiliary verbs such as *has* and *have* ,which appear in sentences like *John has forgotten his umbrella*, are handled using an identity function on et predicates and are labeled V_AUX.

- The lexical items A and SOME are used for analyzing the words *a* and *some* by employing the existential quantification function. The denotation is a function that takes two predicates of type et as arguments (for example *student* and *came*) and returns 1 if and only if an entity that satisfies both predicates exists.

- The lexical item EVERY is used for analyzing sentences such as *every student ran*, by employing the universal quantification function. The denotation is a function that takes two predicates of type et as arguments (for instance *student* and *ran*) and returns 1 if and only if every entity that satisfies the first predicate also satisfies the second.

- THE is used for analyzing the definite article *the*. The denotation is a

[3] For the present purposes the lexicon does not model properties like tense, aspect and person. Thus such information is no represented (see also the treatment of auxiliary verbs). The lexicon can be extended in the future in order to capture inferences that rely on this type of information.

[4] The type of modifiers is explained in Section 4.3.4 under the discussion of M_R and M_I.

function that takes an *et* predicate A and returns the ι operator applied to it. The operator is explained in Section 4.3.5.

- The lexical item EMPTYDET is used for analyzing covert determiners / syntactic operators, as in *Siemens will sell technology* or *John loves ferry tales*, where no article appears before the nouns. The denotation is a function that takes a predicate of type *et* and returns the ζ operator applied to it. The operator is explained in Section 4.3.6.

- The lexical item OF is used for analyzing a preposition that follows a relational noun such as *colleague* in *A colleague of John* and *mother* in *The mother of John.* The function denoted by OF takes a generalized quantifier A of type $(et)t$ (*John*) and a relational noun predicate B of type *eet* (*colleague*), and returns an *et* predicate that characterizes the set of all entities that are in the relation B with A. In the above mentioned example, this translates to the set of entities that are colleagues of John.

- The lexical item POSS is used for analyzing the possessive 's in expressions like *John's colleagues* and *John's mother.* The denotation is a function that takes a generalized quantifier A (*John*) and a relational noun predicate B (*colleagues*) and returns, by the iota operator, an entity that is in the relation B to A. This makes the interpretation of *John's colleagues* and *John's mother* equivalent to *The colleagues of John* and *The mother of John*, respectively.

- A preposition preceding a gerund, as in *by using a wiper*, is analyzed using the lexical item P_GR, which designates the identity function on $(et)et$ predicates.[5]

- The relative pronoun *who* is a functionally ambiguous form that allows noun modification either by a restrictive relative clause or by an appositive clause. The former is an instance of intersective modification, expressed in sentences such as *the [alien [who is a singer]] sat.* In this case the pronoun *who* creates a complex predicate, *alien who is a singer*, from the predicate constituents *alien* and *is a singer.* The function WHO_R that creates this complex predicate for restrictive clauses is the same as the conjunction function AND between *et* functions. The appositive use of the relative pronoun *who* appears in sentences such as *[[the alien], [who is a singer]], sat.* This expresses an appositive modification, as the clause headed by the pronoun adds information regarding a given generalized quantifier B (usually a lifted entity) by adding a condition that it is required to satisfy a given predicate A. In the example above, B is the lifted entity denoted by *the alien* and the predicate A is the denotation of *is a singer.* The resulting generalized quantifier is B if A holds of B, and undefined otherwise. This appositive usage of the relative pronoun is

[5] The type of gerunds is explained in Section 4.3.4 under the discussion of GR_1 and GR_2.

defined using the function WHO_A. Another denotation of WHO_A is used in sentences such as *[[the alien], [Ziggy Stardust]], played guitar*, in which the two components of the appositive construction are noun phrases. In these cases the second phrase in the construction, *Ziggy Stardust*, is interpreted as a predicate, *[who is Ziggy Stardust]*, and the rest of the analysis is equivalent to the former variant of WHO_A.

It should be noted that WHO_A is intended for analyzing only the basic types of appositive modification as illustrated above. This analysis is not suitable for analyzing special appositives, such as locative appositives, since it ignores their idiosyncratic meaning. For example, using WHO_A in *Barack Obama is from Honolulu, Hawaii* would incorrectly render it equivalent to *Barack Obama is from Honolulu, which is Hawaii.*[6]

4.3.4 Template-Based Denotations

In addition to lexical items that are associated with constant denotations, the lexicon also includes items associated with template-based denotations. These are functions that include a place-holder and designed for modeling content words. The place-holder, named *WORD*, is filled when a word is bound to a lexical item, thus fixing the denotation assigned to this word in Σ by the interpretation function I. For example, transitive verbs are analyzed using the lexical item V_2, whose denotation is the template function $\lambda A_{(et)t}.\lambda x_e.A(\lambda y_e.WORD_{eet}(y)$ $(x))$. When a transitive verb such as *eat* is bound to V_2, *eat* is inserted to Σ and I assigns it the denotation $\lambda A_{(et)t}.\lambda x_e.A(\lambda y_e.\mathbf{eat}_{eet}\ (y)(x))$. Template-based denotations are listed in Table 4.4 with their lexical items, types and selected examples, and are explained below.

- Common nouns, such as *girl* and *student*, are analyzed using the lexical item N_1 to denote *et* predicates. For example, the word *girl* bound to N_1 denotes the predicate $\mathbf{girl}_{et}$ – characterizing an arbitrary set of entities that count as a girl. Relational nouns, such as *mother* and *colleague*, are analyzed using N_2 whose denotation is a relation between two entities. For example, by being bound to N_2, *mother* is assigned the denotation $\lambda A_{(et)t}.\lambda x_e.A(\lambda y_e.\mathbf{mother}_{eet}(y)(x))$, whereby $\mathbf{mother}_{eet}$ is a relation between an entity who is a mother and an entity who is her child. The first entity in the denotation is lifted.

- Proper names, such as *Bill* and *Sue*, are analyzed using the lexical item *NP* and treated as lifted entities. For instance, *Bill* is assigned the denotation $\lambda A_{et}.A(\mathbf{bill}_e)$ of type $(et)t$. When this function is applied to a predicate of type *et*, for instance *jumps*, it returns 1 if **bill** is a member of the set of jumpers and 0 otherwise.

[6] The right analysis should treat the sentence as equivalent to *Barack Obama is from Honolulu, which is located in Hawaii.* To support this analysis the lexicon needs to be extended and an arbitrary function $\mathbf{located\_in}_{eet}$ needs to be assumed.

Label	Type	Denotation Template	Example	Remark
N_1	et	$WORD_{et}$	girl	noun
N_2	$((et)t)et$	$\lambda A_{(et)t}.\lambda x_e.$ $A(\lambda y_e.WORD_{eet}(y)(x))$	mother	relational noun
NP	$(et)t$	$\lambda A_{et}.A(WORD_e)$	John	proper name (pn)
V_1	et	$WORD_{et}$	dance	intransitive verb
V_2	$((et)t)et$	$\lambda A_{(et)t}.\lambda x_e.$ $A(\lambda y_e.WORD_{eet}(y)(x))$	eat	transitive verb
FACT	tet	$\lambda c_t.\lambda z_e.WORD_{tet}(c)(z) \wedge c$	realize	factive verb
NOFACT	tet	$\lambda c_t.\lambda z_e.WORD_{tet}(c)(z)$	think	non-factive verb
MI	$(et)et$	$\lambda A_{et}.\lambda x_e.A(x) \wedge WORD_{et}(x)$	Dutch	intersective modifier
		$\lambda A_{(et)t}.\lambda B_{et}.A(B)\wedge$ $A(WORD_{et})$		intersective modifier of pn
MR	$(et)et$	$\lambda A_{et}.\lambda x_e.A(x)\wedge$ $WORD_{(et)et}(A)(x)$	tall	restrictive modifier
NR	$(et)et$	$\lambda A_{et}.\lambda x_e.WORD_{(et)et}(A)(x)$	fake	non-restrictive modifier
P_R	$((et)t)(et)et$	$\lambda A_{(et)t}.\lambda B_{et}.\lambda x_e.(B(x)\wedge$ $(A(\lambda y_e.WORD_{e(et)et}(y)(B)(x)$	in	restrictive preposition
P_I	$((et)t)(et)et$	$\lambda A_{(et)t}.\lambda B_{et}.\lambda x_e.(B(x)\wedge$ $(A(\lambda y_e.WORD_{eet}(y)(x)$		intersective preposition
GER_1	$(et)et$	$\lambda A_{et}.\lambda x_e.A(x)\wedge$ $WORD_{(et)et}(A(x))$	walking	intransitive gerund
GER_2	$((et)t)(et)et$	$\lambda A_{(et)t}.\lambda B_{et}.\lambda x_e.B(x)\wedge$ $A(\lambda y_e.WORD_{e(et)et}(y)(B)(x))$	visiting	transitive gerund
NUMBER	$(et)(et)et$	$\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.$ $WORD_{et}(x) \wedge A(x) \wedge B(x)$	five	number

Table 4.4: Templates of Denotations in the Interpreted Lexicon

- Verbs are analyzed similarly to nouns. The lexical item V_1 is used for binding intransitive verbs such as *dance* and the lexical item V_2 for transitive verbs such as *eat*.[7]

- Transitive verbs that take a clausal argument are analyzed based on the interpretation of their argument as either factual or non-factual. The lexical item FACT is used for verbs with a factual argument, such as *realized* in *John realized that the train has left*, which entails that *the train has left.* The lexical item NOFACT is used for verbs with a non-factual argument, such as *thinks* in *John thinks that the train has left*, which does not entail that *the train has left.* Binding the word *realized* to FACT assigns it the denotation $\lambda c_t.\lambda z_e.\mathbf{realized}_{tet}(c)(z) \wedge c$, which takes an argument c of type t – the clausal argument – and returns a predicate of type et. This predicate is satisfied for entities that are in an arbitrary binary relation – in the case in point, **realized** – to c and requires that c is 1. By contrast, NOFACT does not require c to be 1. Non-factual arguments are not necessarily true.
  It should be noted that this basic analysis of factives is only intended for supporting simple inferences as illustrated above. It is assumed that the subordinate clause is a self-contained clause whose truth-value is calculated independently of the matrix clause. But this is not always the case, for example: *Every man who has a donkey realizes that it is a cute animal.* In addition, the analysis ignores the presuppositional nature of factives.

- When modifying nouns or verbs, most modifiers restrict the denotation of the noun/verb to which they attach. Thus, whether John is a *Dutch swimmer* or a *short swimmer*, he is necessarily a *swimmer*. But unlike *short*, the adjective *Dutch* is also intersective in the following sense: a *Dutch swimmer* is invariably *Dutch*, while a *short swimmer* is not necessarily *short*, relatively speaking. An intersective modifier is assigned a denotation by binding it to the lexical item MI, while a restrictive non-intersective modifier is bound to MR. Binding *Dutch* to MI results in its assignment to the function $\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{dutch}_{et}(x)$ – with the embedded arbitrary constant **dutch** of type et characterizing all the Dutch entities. The resulting function requires entities to satisfy both the noun predicate and the predicate **dutch**. Binding *short* to MR assigns to this word the denotation $\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{short}_{(et)et}(A)(z)$, which requires entities to satisfy the noun predicate A and also its modified version – $\mathbf{short}_{(et)et}(A)$. The latter is created by an arbitrary function **short** of

[7] The presented lexicon does not distinguish between verbs and nouns of the same valency. Thus *jump* is analyzed exactly as *student*, and *eat* as *mother*. This is because linguistic properties that distinguish between verbs and nouns are not modeled. For example, lexical semantic properties such as *event structure*, and syntactic properties such as *tense* in the case of verbs vs. *gender* in the case of nouns. As mentioned before, the lexicon can be extended in the future to model more linguistic properties and to account for inferences related to them.

type $(et)(et)$. This function takes a predicate characterizing a set of entities and returns a predicate characterizing only those that are considered short. This analysis of intersective and restrictive modifiers is the same for all modified predicates of type et – for example, intransitive verbs as in *Laura [danced gracefully]* and transitive verbs as in *John [[ate the cake] quickly]*.

- Another version of intersection is included for modified entities, as in *German Siemens sells medical equipment*. In this case, binding the modifier *German* to MI assigns to it the denotation $\lambda A_{(et)t}.\lambda B_{et}.A(B) \wedge A(\mathbf{german}_{et})$. This function is a generalized quantifier modifier: it takes a generalized quantifier A and returns it with the restriction that the arbitrary function **german** satisfies it. Thus, when applied to a lifted entity, the requirement is that the entity is in the set characterized by **german**.

- Another type of modifiers treated in the lexicon is non-restricting, such as *alleged* and *faked*. These are adjectives that do not restrict the set characterized by the noun that they modify: an *alleged killer* is not necessarily a *killer*, and a *faked frog* is not necessarily a *frog*.[8] The lexical item NR is used for analyzing these modifiers. When a word like *alleged* is bound to NR it is assigned the denotation $\lambda A_{et}.\lambda x_e.\mathbf{alleged}_{(et)et}(A)(z)$ – with an arbitrary constant **alleged** of type $(et)et$ that takes a predicate characterizing a set of entities and returns a predicate characterizing a set of alleged ones.

- Prepositions are analyzed using the lexical items P_R and P_I. P_R is used for instances of restrictive non-intersective modification, as in *John is a [student [in New York]]*, which entails that *John is a student* but not that *John is in New York*. P_I is used for instances of intersective modification, as in *Google is a [[search engine] [on the web]]*, which entails that *Google is a search engine* and also that *Google is on the web*. When *in* is bound to P_R, it is assigned a denotation that takes a generalized quantifier A (*New York*) and an et predicate B (*student*), and returns a modified predicate that an entity x satisfies only if this entity satisfies B and is in a triadic relation, **in**, to A and to all entities satisfying B. In the case of P_I, the entity is only required to be in a binary relation to the lifted entity. Note that, under this analysis, all prepositional phrases analyzed using P_R and P_I are treated as adjuncts of the predicate that they attach to rather than as complements. As explained above, the lexical item OF is used for treating prepositional phrases as complements in the nominal domain.[9]

[8] Furthermore, a *faked frog* is necessarily not a *frog*. For this reason *faked* is also considered a co-restrictive modifier. At this point the lexicon does not make this distinction between different kinds of non-restricting modifiers.

[9] The lexical item OF is restricted to the nominal domain, e.g. *The mother of John*. Support for prepositional complements in the verbal domain can be added in the future.

- Gerunds are analyzed as restrictive modifiers of verbs. The lexical item GER_1 is used for intransitive gerunds, such as *hiking* in *Bob spent the weekend hiking*, and GER_2 is used for transitive gerunds, such as *drinking* in *John waited for a flight drinking an orange juice.* GER_1 is identical to MR, which means that it denotes a modifier of an *et* predicate. Thus, when *hiking* is bound to GER_1, it is assigned the denotation $\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{hiking}_{(et)et}(A(x))$, which takes an *et* predicate A (*spent the weekend*) and returns a predicate that is satisfied only for entities that satisfy A and the restriction of A to those who hike. GER_2 is similar but also encompasses the argument of the gerund. Accordingly, binding *drinking* to GER_2 assigns it the denotation $\lambda A_{(et)t}.\lambda B_{et}.\lambda x_e.B(x) \wedge A(\lambda y_e.\mathbf{drinking}_{e(et)et}(y)(B)(x))$. This function takes the gerund's argument – a generalized quantifier A (*an orange juice*) – and the *et* predicate B that the gerund modifies (*waited for a flight*), and returns a modified predicate of type *et*. This predicate requires entities to satisfy B and also to be in a triadic relation, **drinking**, to A and to the set of entities characterized by B.

- Numbers such as *twenty*, in *twenty students passed the exam*, are analyzed similarly to existential quantifiers. Binding the word *twenty* to NUMBER assigns it with denotation $\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{twenty}_{et}(x) \wedge A(x) \wedge B(x))$. This function takes two *et* predicates, A (*students*) and B (*passed the exam*), and returns 1 if and only if there is an entity that satisfies both of them as well as an arbitrary *et* predicate, **twenty**. This analysis allows to infer that *some students passed the exam* follows from *twenty students passed the exam* and not vice versa. However, no arithmetic relation between quantities can be inferred from it.

4.3.5 The Iota Operator

The ι operator describes definite descriptions semantically. It is employed in the denotations of THE and WHO_A for analyzing the definite article and appositive constructions. The analysis adopted in this model borrows ideas from Strawson (1950), von Fintel (1999) and Szabó (2000), among others.

The operator is a function of type $(et)e$. It takes a predicate of type *et* as an argument and returns an entity from the predicate's characteristic set. In addition, it is assumed that definite descriptions refer only to entities that exist and uniquely satisfy a certain property. For example, the definite description in *John knows the man who kissed Mary* refers to an entity from the extension of the complex predicate $[\![\textit{man who kissed Mary}]\!]^M$. For the sentence to be felicitous, this entity has to exist and to be the only one who is a man that kissed Mary. In set-theory terms, these putative properties of existence and uniqueness require the characteristic set of the complex predicate to be a singleton.

Following Strawson (1950), we assume that the uniqueness requirement is a presupposition of the sentence. Thus, in *John knows the man who kissed Mary*, the sentence presupposes that there is a unique man who kissed Mary in the context in which the sentence is interpreted. This means that a mechanism of presupposition projection defines the uniqueness as a requirement from the context of a sentence rather than as part of its truth-conditions. If a presupposition fails, for instance in a model where two man kissed Mary, the sentence in not interpretable.

In line with von Fintel's (1999) concept of *Strawson Entailment*, we assume that presuppositions projected from the hypothesis are in the same status as those projected from the text.[10] Therefore, all uniqueness requirements from both text and hypothesis are phrased as requirements from the context in which the entailment is assessed. We use the term *entailment context* to refer to this notion of context.

This treatment of presuppositions is supported by judgments of native speakers. Speakers indicate that pairs such as (29) and (30) are naturally interpreted as positive entailment pairs.

(29) *t*: The man who kissed Mary jumped.

h: The man jumped.

(30) *t*: A man who kissed Mary jumped.

h: The man who kissed Mary jumped.

Fundamentally, these judgements indicate that the simpler inference in (31) is treated as valid.

(31) *t*: A man jumped.

h: The man jumped.

The idea is that understanding an entailment as a relation between a text and hypothesis brings about a combined context in which definite descriptions are interpreted. Therefore, a definite description in the hypothesis can refer to an (in)definite description in the text while keeping the entailment relation valid.

Notice, however, that an entailment pair as in (32) is naturally interpreted as a negative entailment pair. We take this as an indication that the existence requirement associated with definite descriptions should not be treated as a presupposition. For if it was, an entity that satisfies **man** and **kiss**(**m**), assuming **m** is the denotation of Mary, would have been presupposed and the non-intuitive entailment in (32) would have followed.

[10] von Fintel (1999) is mainly concerned with presuppositions projected from generalized quantifiers such as *Only John*. We embrace the same treatment for presuppositions projected from definite descriptions.

(32) *t*: The man jumped.

h: The man who kissed Mary jumped.

Due to that we analyze existence requirements as part of the truth-conditions of the sentence in which they appear. This puts the uniqueness and existence requirements in different scopes of interpretation: uniqueness is global because it is taken to be part of the context of the entailment assessment, while existence is local because it is treated as part of the truth-conditions of a sentence. This kind of distinction is in line with recent proposals such as Szabó (2000) and Ludlow and Segal (2004).

Formally, we indicate the truth-conditions of a sentence S by TC(S) and its presuppositions by PS(S). The combination of PS(t) and PS(h) constitutes the entailment context. For assessing the entailment in a *t*-*h* pair we define two formulas:

- $formula1 = \text{TC}(t) \wedge \text{PS}(t) \wedge \text{PS}(h)$
- $formula2 = \text{TC}(h)$

And we check whether $formula1$ logically entails $formula2$, as defined by the truth-conditionality criterion. In the reminder of this thesis we will use the terms "t entails h" for clarity, but in practice this translates to "$formula1$ entails $formula2$". For illustration, (29) is analyzed as shown in (33).

(33) *t*: $(\forall_{(et)t}(\lambda x.\mathbf{man}(x) \wedge \forall_{(et)t}(\lambda y.\mathbf{man}(y) \rightarrow =_{eet} (x)(y)))) \wedge \exists_{(et)t}(\lambda x.\ \mathbf{man}(x) \wedge \mathbf{jump}(x))$

h: $\exists_{(et)t}(\lambda x.\mathbf{man}(x) \wedge \mathbf{jump}(x))$

Thus, the entailment follows.

The logical framework described in this chapter is a bi-valued system. That is, each sentence denotes either 1 or 0. Truth-value gaps are avoided and therefore the uniqueness requirements are treated as a bi-valued propositions. As a result, a presupposition failure results in $formula1$ denoting 0, which then allows to infer any hypothesis. For example, consider the non-coherent entailment in (34).

(34) *t*: The tallest man, John, is a friend of the shortest man, Bill, and John is not Bill.

h: The man is Bill.

The presupposition projected from the hypothesis requires a unique man. Since both John and Bill satisfy *man* but denote different entities, we get that TC(t) contradicts PS(h). Therefore $formula1 = 0$ and any $formula2$ would follow from it.

However, although this is an undesired result, in the context of natural entailment data such as the data in the RTE corpora, this is mostly a theoretical problem that has no practical consequences. The reason is that in the construction of the RTE datasets, sentences were assumed to be coherent. Presupposition failures, which traditionally mean that a sentence is uninterpretable, were not expected to occur in the annotated data. If modelling presupposition failures is required, the model can be extended into a three-valued system (Schlenker, 2008, 2009).

It should be noted here that our analysis of presuppositions as propositions is subject to a technical problem when a presupposition includes a free variable. For example, consider (35), taken from Van der Sandt (1992).

(35) Every man kissed the girl who loved him.

In this case the presupposition projected from the definite description *the girl who loved him* includes a free variable bound by the quantifier *every* in the matrix clause. Van der Sandt (1992) proposes an analysis of presuppositions as anaphoras and defines a free variable trapping constraint that prevents variables from ending up free in DRS structures. Our analysis is more basic and does not deal with problems of this sort.

The last point regarding the proposed analysis concerns the nature of projection of uniqueness presuppositions. The limited scope of linguistic phenomena that the current framework covers allows us to adopt an approach by which presuppositions are projected from any semantic/syntactic environment. We do not employ Karttunen's holes-plugs-filters typology (1973) or more modern proposals (Chierchia and McConnell-Ginet, 2000, Beaver, 2001, Tonhauser et al., 2013), which aim to explain how a sentence like (36a), quoted from Potts (2014), projects a presupposition that incorporates the past time-span (36b) rather then the present (36c).[11]

(36) a. In 1914, Princip assassinated the Archduke.

b. There was a unique Archduke in 1914

c. There is a unique Archduke.

In our model, uniqueness requirements of definite descriptions project regardless of their environment. The model can be extended once more complex phenomena is added.

For the same reason, the current analysis licenses the inference in (37) even if the word *imagined* is annotated as non-factive (NOFACT) as it should be.

[11] Notice that in the case of (36a) no problem appears for the current model because tense and temporal relations are not modeled in the framework. Due to that, the sentences *There was a unique Archduke* and *There is a unique Archduke* are treated (incorrectly by-design) as meaning the same.

(37) *t*: John imagined that Mary loves the boy who loves Sue.

h: A boy loves Sue.

The presupposition projected from the definite description *the boy who loves Sue* in t allows to derive h. A similar pattern appears in (38), where the presupposition is projected from the embedded appositive modification.

(38) *t*: John imagined that Mary, who is a tall girl, loves Bill.

h: Mary is a tall girl.

As in (37), this inference is licensed regardless of the annotation of the word *imagined* as factive or non-factive.[12]

4.3.6 The Zeta Operator

This operator takes an *et* predicate and returns an entity that serves for representing the intensional class for the predicate in the semantic analysis. The ζ operator is used in noun phrases like the following, which miss an over determiner expression.

- Fairy tales are enjoyable.
- John loves short stories.
- Siemens sells medical equipment.

Consider for example the sentence *Fairy tales are enjoyable*, which describes the nature of fairy tales. In such cases adding an empty determiner results in applying the ζ operator to the plural noun fairy tales. The result is an entity describing the class of fairy tales. Carlson (1977), who initiated this analysis, refers to representative entity such as this one as *kinds*. Note that a kind entity is not a member of the set it represents. For example, the kind for *fairy tales* is not among the entities in the extension of the predicate **fairy_tales**.

In many cases a sentence is ambiguous between an existential reading and a kind or a generic reading. For example, *medical equipment* in *Siemens sells medical equipment* can be understood as bound existentially, that is *Siemens sells some medical equipment*, or as bound by some generic operator, as part of a description of the habitual behavior of Siemens. If the former interpretation predominates then an existential determiner is to be added. If the latter, an empty determiner that employs the ζ operator is used.

[12] This kind of examples may give rise to de-re/de-dicto ambiguities. On the de-re reading, Mary is a tall girl such that John imagined that she loves Bill. The inference that Mary is a tall girl follows directly from this reading. On the de-dicto reading the modality operator conveyed by *imagined* takes scope over the apposition *Mary, a tall girl.* In such cases the inference would not follow.

4.4 A Semantic Model of Entailment with Lexical Information

We have so far described the theoretical concept of entailment in a semantic model and explained the treatment of certain linguistic phenomena encoded in the lexicon. In this section we show how the proposed theory constitutes a general semantic model of entailment that can be used to analyze pairs from the RTE.

4.4.1 A Basic Model

Consider the following three-step analysis:

1. Phrase-Structure Analysis: the raw text-hypothesis data are analyzed using binary constituency-based phrase structures.
2. Binding to Lexicon: words and phrases are bound to lexical items from the interpreted lexicon.
3. Proof of Entailment: using lambda calculus reductions, a logical proof is established that the hypothesis follows from the text.

Let us apply this analysis to the following example:

(39) *t*: The book contains short stories by the famous Bulgarian writer Nikolai Haitov.

h: Nikolai Haitov is a writer.

Accordingly:

1. Phrase-Structure Analysis:

   t: [The book] [contains [[short stories] [by [[the [famous [Bulgarian writer]]] [Nikolai Haitov]]]]].

   h: [Nikolai Haitov] [is [a writer]].

2. Binding to Lexicon:

   t: [The/THE book/N_1] [contains/V_2 [short/MR stories/N_1] [by/P_R [[the/THE [famous/MR [Bulgarian/MR writer/N_1]]] [Nikolai Haitov]/NP]]]

   h: [Nikolai Haitov]/NP [is/IS [a/A writer/N_1]].

   Most of the bindings in this example are self-evident: the definite articles are bound to THE, the nouns *book*, *stories* and *writer* are bound to N_1, *contains* is bound to V_2 since it is a transitive verb, *by* is bound to the preposition lexical item P_R, *a* to A and *is* to IS.

The title construction *the famous Bulgarian author Nikolai Haitov* is analyzed as an appositive, to express that Nikolai Haitov *is* the famous Bulgarian author. Since there is no specific word to bind to WHO_A, a token element, *APP* is added and bound correspondingly.

In addition, the modified noun *short stories by...* which appears bare, without any quantificational element, needs to be addressed. The intuitive interpretation of the text is that the book contains *some of the* short stories by Haitov. The existential quantifier is not included in the text but it still plays a role in the intuitive interpretation of this sentence. Therefore a determiner bound to SOME is added to the phrase structure as part of the annotation process.[13]

The bindings that require further explanation are those of the modifiers *famous* and *Bulgarian*. These are the only bindings that require examination of the inference between t and h. The two adjectives are bound to MR because the only modification property that they must satisfy in this example is restrictivity: *famous Bulgarian writer* → *writer*. If, however, the hypothesis was phrased as: *Nikolai Haitov is famous* or *Nikolai Haitov is Bulgarian*, then the property of intersectivity, which licenses these inferences, had to be indicated by binding the modifiers: *Bulgarian* and *famous* in the former case and *Bulgarian* in the latter, to MI.

This nuance is particularly salient with respect to ethnic adjectives: sometimes the inference requires that they should be interpreted as restrictive but at the same time does not rule out an intersective interpretation, as the pair above illustrates; so both MR and MI bindings are possible for *Bulgarian*. Sometimes the inference requires that ethnic adjectives should be interpreted as intersective, since restrictivity alone does not account for the inference, as illustrated by the artificial hypothesis *Nikolai Haitov is Bulgarian*: only MI is possible for *Bulgarian* in this case. In other instances, however, the inference requires a restrictive interpretation and rules out an intersective one: *My daughter takes French ballet classes* entails that *My daughter takes ballet classes* but, intuitively, not that *My daughter takes French classes*. Binding *French* to MR is the only option to support the former and prevent the latter.

The word *short* is annotated as a restrictive modifier of *stories* although the entailment does not rely on the modification in the phrase *short stories*. This is because the lexicon offers only a restrictive or an intersective annotation for an adjective, and the restrictive one is weaker in terms of the inferences it licenses.[14]

[13] Note that the addition of a determiner is required also type-wise: the modified noun denotes a function of type et, while the verb *contains* is bound to V_2, which is assigned a denotation of type $((et)t)et$. None of these elements is a function that takes the other as an argument.

[14] As explained in Section 3.2, every intersective modifier is restrictive but not vice versa.

The phrase structure at the end of this stage is as follows:

t: [The/THE book/N_1] [contains/V_2 [some/SOME [[short/MR stories/N_1] [by/P_R [[the/THE [famous/MR [Bulgarian/MR writer/N_1]]]APP/WHO_A [Nikolai Haitov]/NP]]]]]

h: [Nikolai Haitov]/NP [is/IS [a/A writer/N_1]].

3. Proof of Entailment:
Let M be an intended model,

$[\![$[The/THE book/N_1] [contains/V_2 [some/SOME [[short/MR stories/N_1] [by/P_R [[the/THE [famous/MR [Bulgarian/MR writer/N_1]]]APP/WHO_A [Nikolai Haitov]/NP]]]]]$]\!]^M$

$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\textsc{who\_a}([\mathit{Nikolai\ Haitov}]/\textsc{np}))(\mathit{the}/\textsc{the}$ $(\mathit{famous}/\textsc{mr}\ (\mathit{Bulgarian}/\textsc{mr}(\mathit{writer}/\textsc{n\_1})))))))))$ $(\mathit{the}/\textsc{the}(\mathit{book}/\textsc{n\_1}))$	analysis
$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\textsc{who\_a}(\lambda A_{et}.A(\mathbf{h}_e)))(\lambda A_{et}.\lambda B_{et}.B(\iota(A))$ $(\mathit{famous}/\textsc{mr}(\mathit{Bulgarian}/\textsc{mr}(\mathbf{writer})))))))))$ $(\lambda A_{et}.\lambda B_{et}.B(\iota(A))(\mathbf{book}))$	denotations of NP, THE, N_1, V_2
$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\textsc{who\_a}(\lambda A_{et}.A(\mathbf{h}_e)))(\lambda B_{et}.B(\iota(\mathit{famous}/\textsc{mr}$ $(\mathit{Bulgarian}/\textsc{mr}(\mathbf{writer})))))))))(\lambda B_{et}.B(\iota(\mathbf{book})))$	func. app. to $\iota(\mathit{famous}\ldots)$, **book**
$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\lambda A_{(et)t}.\lambda B_{(et)t}.\lambda C_{et}.C(\iota(\lambda x_e.B(=(x)) \wedge A(=$ $(x))))(\lambda A_{et}.A(\mathbf{h}_e)))(\lambda B_{et}.B(\iota(\mathit{famous}/\textsc{mr}(\mathit{Bulgarian}$ $/\textsc{mr}(\mathbf{writer})))))))))(\lambda B_{et}.B(\iota(\mathbf{book})))$	denotation of WHO_A
$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\lambda B_{(et)t}.\lambda C_{et}.C(\iota(\lambda x_e.B(=(x)) \wedge =(x)(\mathbf{h}_e))))$ $(\lambda B_{et}.B(\iota(\mathit{famous}/\textsc{mr}(\mathit{Bulgarian}/\textsc{mr}(\mathbf{writer})))))))))$ $(\lambda B_{et}.B(\iota(\mathbf{book})))$	func. app. to $\lambda A_{et}. A(\mathbf{h}_e)$, $=(x)$
$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\lambda C_{et}.C(\iota(\lambda x_e.(=\ (x)(\iota(\mathit{famous}/\textsc{mr}$ $(\mathit{Bulgarian}/\textsc{mr}(\mathbf{writer})))))\wedge\ =\ (x)(\mathbf{h}_e))))))))$ $(\lambda B_{et}.B(\iota(\mathbf{book})))$	func. app. to $(\lambda B_{et}. B(\iota(\mathit{famous}\ldots,$ $=(x)$

$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\lambda C_{et}.C(\iota(\lambda x_e.(= (x)(\iota(\lambda A_{et}.\lambda z_e.A(z) \wedge$ $\mathbf{famous}_{(et)et}(A)(z)(\lambda A_{et}.\lambda z_e.A(z) \wedge \mathbf{bulgarian}_{(et)et}$ $(A)(z)(\mathbf{writer})))))\wedge = (x)(\mathbf{h}_e))))))))(\lambda B_{et}.B$ $(\iota(\mathbf{book})))$	denotation of $\textsc{mr}$
$= (\mathit{contains}/\textsc{v\_2}(\mathit{some}/\textsc{some}((\mathit{short}/\textsc{mr}(\mathit{stories}/\textsc{n\_1}))$ $(\mathit{by}/\textsc{p\_r}((\lambda C_{et}.C\ (\iota(\lambda x_e.(= (x)(\iota(\lambda z_e.(\mathbf{writer}(z)\wedge$ $\mathbf{bulgarian}_{(et)et}\ (\mathbf{writer})(z) \wedge \mathbf{famous}_{(et)et}(\lambda y_e.$ $\mathbf{writer}(y) \wedge \mathbf{bulgarian}_{(et)et}(\mathbf{writer})(y))(z)\ )))\wedge =$ $(x)(\mathbf{h}_e)))))))\ (\lambda B_{et}.B(\iota(\mathbf{book})))= (*)$	func. app. to **writer**, **bulgarian**, **famous**

The expression in (*) contains three applications of the iota operator that correspond to three definite descriptions:

(a) $\iota(\lambda z_e.(\mathbf{writer}(z) \wedge (\mathbf{bulgarian}_{(et)et}(\mathbf{writer}))(z) \wedge \mathbf{famous}_{(et)et}$ $(\lambda y_e.\mathbf{writer}(y)\ \wedge(\mathbf{bulgarian}_{(et)et}(\mathbf{writer}))(y))(z))$
(*The famous Bulgarian writer*)

(b) $\iota(\mathbf{book})$
(*The book*)

(c) $\iota(\lambda x_e.(= (x)(\iota(\lambda z_e.(\mathbf{writer}(z)\wedge\ \mathbf{bulgarian}_{(et)et}(\mathbf{writer})(z) \wedge$ $\mathbf{famous}_{(et)et}(\lambda y_e.\ \mathbf{writer}(y)\wedge\mathbf{bulgarian}_{(et)et}(\mathbf{writer})(y))(z)\ )))\wedge =$ $(x)(\mathbf{h}_e)))$
(*The entity who is both the famous Bulgarian writer and Nikolai Haitov*)

As explained in Section 4.3.5, it is assumed that such definite descriptions carry a uniqueness requirement that is taken to be part of the entailment context and an existence requirement. The entailment context is thus the conjoined uniqueness requirements:
$Ctx =$
$[\forall_{(et)t}(\lambda a_e.(\mathbf{writer}(a) \wedge (\mathbf{bulgarian}_{(et)et}(\mathbf{writer}))(a) \wedge \mathbf{famous}_{(et)et}$ $(\lambda y_e.\mathbf{writer}(y) \wedge (\mathbf{bulgarian}_{(et)et}(\mathbf{writer}))(y))(a) \wedge \forall_{(et)t}(\lambda b_e.\mathbf{writer}$ $(b) \wedge (\mathbf{bulgarian}_{(et)et}(\mathbf{writer}))(b) \wedge \mathbf{famous}_{(et)et}(\lambda y_e.\mathbf{writer}(y)\wedge$ $(\mathbf{bulgarian}_{(et)et}(\mathbf{writer}))(y))(b) \rightarrow= (a)(b)))] \wedge$

$[\forall_{(et)t}(\lambda a_e.\mathbf{book}(a) \wedge \forall_{(et)t}(\lambda b_e.\mathbf{book}(b) \rightarrow= (a)(b)))] \wedge$

$[\forall_{(et)t}(\lambda a_e.(= (a)(\iota(\lambda z_e.(\mathbf{writer}(z) \wedge \mathbf{bulgarian}_{(et)et}(\mathbf{writer})(z)]\wedge$ $\mathbf{famous}_{(et)et}(\lambda y_e.\mathbf{writer}(y) \wedge \mathbf{bulgarian}_{(et)et}(\mathbf{writer})(y))(z))))\wedge$ $= (a)(\mathbf{h}_e)) \wedge \forall_{(et)t}(\lambda b_e. = (b)(\iota(\lambda z_e.(\mathbf{writer}(z) \wedge \mathbf{bulgarian}_{(et)et}$ $(\mathbf{writer})(z) \wedge \mathbf{famous}_{(et)et}(\lambda y_e.\mathbf{writer}(y) \wedge \mathbf{bulgarian}_{(et)et}$ $(\mathbf{writer})(y))(z))))\wedge = (b)(\mathbf{h}_e) \rightarrow= (a)(b)))]$

The existence requirements are combined into the following formula:
$\exists_{(et)t}(\lambda a_e.(\mathbf{writer}(a) \wedge \mathbf{bulgarian}_{(et)et}(\mathbf{writer})(a) \wedge \mathbf{famous}_{(et)et}$

$(\lambda y_e.\mathbf{writer}(y) \wedge \mathbf{bulgarian}_{(et)et}\ (\mathbf{writer})(y))(a)) \wedge \exists_{(et)t}(\lambda b_e.$ $\mathbf{book}(b) \wedge \exists_{(et)t}(\lambda c_e.(=(c)(a) \wedge =(c)(\mathbf{h}_e))$

We can continue the proof from (*):

$$\begin{aligned}
&= Ctx \wedge \exists_{(et)t}(\lambda a_e.(\mathbf{writer}(a) \wedge (\mathbf{bulgarian}_{(et)et} \\
&\quad (\mathbf{writer}))(a) \wedge \mathbf{famous}_{(et)et}(\lambda y_e.\mathbf{writer}(y) \wedge \\
&\quad \mathbf{bulgarian}_{(et)et}\ (\mathbf{writer})(y))(a)) \wedge \exists_{(et)t}(\lambda b_e.\mathbf{book} \\
&\quad (b) \wedge \exists_{(et)t}(\lambda c_e.(= (c)(a) \wedge = (c)(\mathbf{h}_e)) \wedge \\
&\quad (contains/\textsc{v\_2}(some/\textsc{some}((short/\textsc{mr}(stories/\textsc{n\_1})) \\
&\quad (by/\textsc{p\_r}(\lambda C_{et}.C(c))))))(\lambda B_{et}.B(b))))) && \text{treatment of iotas} \\
&\leq \exists_{(et)t}(\lambda a_e.(\mathbf{writer}(a) \wedge \exists_{(et)t}(\lambda c_e.(= (c)(a) \wedge = \\
&\quad (c)(\mathbf{h}_e))))) && \text{def. of } \wedge \\
&= \exists_{(et)t}(\lambda a_e.\mathbf{writer}(a) \wedge = (a)(\mathbf{h}_e)) && \text{transitivity of } = \\
&= \exists_{(et)t}(\lambda a_e.\mathbf{writer}(a) \wedge (\lambda A_{et}.A(\mathbf{h}_e))(\lambda z_e. = (a)(z))) && \text{func. app.} \\
&= (\lambda Q_{(et)t}.\exists_{(et)t}(\lambda a_e.\mathbf{writer}(a) \wedge Q(\lambda z_e. = (a)(z)))) \\
&\quad (\lambda A_{et}.A(\mathbf{h}_e)) && \text{func. app.} \\
&= (\lambda Q_{(et)t}.\exists_{(et)t}(\lambda a_e.\mathbf{writer}(a) \wedge (\lambda y_e.Q(\lambda z_e. = (y)(z))) \\
&\quad (a))(\lambda A_{et}.A(\mathbf{h}_e)) && \text{func. app.} \\
&= (\lambda Q_{(et)t}.(\lambda B_{et}.\exists_{(et)t}(\lambda a_e.\mathbf{writer}(a) \wedge B(a)))(\lambda y_e.Q \\
&\quad (\lambda z_e. = (y)(z))))(\lambda A_{et}.A(\mathbf{h}_e)) && \text{func. app.} \\
&= ((\lambda P_{(et)t}.\lambda Q_{(et)t}.P(\lambda y_e.Q(\lambda z_e. = (y)(z))))(\lambda B_{et}. \\
&\quad \exists_{(et)t}(\lambda a_e.\mathbf{writer}(a) \wedge B(a))))(\lambda A_{et}.A(\mathbf{h}_e)) && \text{func. app.} \\
&= ((\lambda P_{(et)t}.\lambda Q_{(et)t}.P(\lambda y_e.Q(\lambda z_e. = (y)(z))))(\lambda A_{et}. \\
&\quad \lambda B_{et}.\exists_{(et)t}(\lambda a_e.A(a) \wedge B(a)))(\mathbf{writer}))(\lambda A_{et}.A(\mathbf{h}_e)) && \text{func. app.} \\
&= (\textsc{is}_{eq}(\textsc{a}(\mathbf{writer})))(\lambda A_{et}.A(\mathbf{h}_e)) && \text{denotations of } \textsc{is}_{eq},\ \textsc{a} \\
&= (is/\textsc{is}_{eq}(a/\textsc{a}(writer/\textsc{n\_1})))([Nikolai\ Haitov]/\textsc{np}) && \text{denotations of } \textsc{n\_1},\ \textsc{np}
\end{aligned}$$

$$= \llbracket\ \mathit{[Nikolai\ Haitov]/\textsc{np}(is/\textsc{is}_{eq}(a/\textsc{a}(writer/\textsc{n\_1}))}\ \rrbracket^M$$

This result shows that the formal model provides a theoretical explanation for the natural inference in (39). To summarize, this result is achieved through a phrase-structure analysis followed by binding words and phrases to the interpreted lexicon, which in turn is succeeded by a theoretical proof of the entailment by means of lambda calculus reductions.

4.4.2 A Model with Lexical Information

In the analysis we described so far, all lexical information is provided by the logical terms of entries as defined in the lexicon. The lexical semantics of words is not represented and due to that lexical relations between word meanings are not modeled. However, such information is critical for restricting the intended models in which an entailment relation is evaluated. For example, consider the pair in (40):

(40) *t*: The head of the Italian opposition Romano Prodi, was the last president of the European Commission.

h: Romano Prodi is a former president of the European Commission.

In this example, the relation between the adjectives *last* and *former* is required for licensing the entailment. Formally this translates to a restriction on the set of intended models in which the truth-conditionality criterion is evaluated. Only models where the relation $\forall A_{et}.\forall x_e.last_{(et)et}(A)(x) \rightarrow former_{(et)et}(A)(x)$ holds count.[15]

We now introduce our complete analysis and treat these relations as part of the background assumptions posited for the purpose of proving the entailment. The *entailment context* described above in relation to definite descriptions is most suitable for maintaining this information. Consider the following four-step analysis:

1. Phrase-Structure Analysis: the text and hypothesis are analyzed using binary constituency-based phrase structures.
2. Binding to Lexicon: words and phrases are bound to lexical items from the interpreted lexicon.
3. Expressing lexical relations: lexical inferences between words/phrases in the text and hypothesis are expressed as non-logical axioms.
4. Proof of Entailment: using lambda calculus reductions, a logical proof is established that the hypothesis follows from the text. The lexical relations expressed in the previous step are treated as part of the assumptions.

[15] We define relations that are sufficient for capturing the lexical inference in the entailment at hand but they may not be correct for other examples. For instance, if Prodi is the current president of the EC, then the sentence *Prodi is the last president of the EC* does not entail that he is a former president of the EC. This is because the relation as indicated above ignores the tense of the sentence. Describing relations in a way that makes them universally true requires a more comprehensive semantic theory.

Let us now apply these four steps to example (40):

1. Phrase-Structure Analysis:

   t: [[The [head [of [the [Italian opposition]]]]], [Romano Prodi]] [was [the [last [president [of [the [European Commission]]]]]]].

   h: [Romano Prodi] [is [a [former [president [of [the [European Commission]]]]]]].

2. Binding to Lexicon:

   t: [[The/THE [head/N_2 [of/OF [the/THE [Italian/MR opposition/N_1]]]]] APP/WHO_A [Romano Prodi]/NP], [was/IS$_{eq}$[the/THE [last/NR [president/N_2 [of/OF [the/THE [European Commission]/N_1]]]]]].

   h: [Romano Prodi]/NP [is/IS [a/A [former/NR [president/N_2 [of/OF [the/THE [European Commission]/N_1]]]]]].

   The annotations in this example are straightforward: the definite articles are bound to THE, the noun *opposition* is bound to N_1 and so is the compound *European Commission* because it is preceded by *the*, the relational nouns *head* and *president* are bound to N_2, and the proper name *Romano Prodi* is bound as a phrase to NP. The adjective *Italian* is bound to MR, as it is naturally understood as a restrictor of the extension of the noun *opposition*. Thus, although the annotations of MI and NR are also possible for this adjective, in the context of this entailment, allowing an inference of *Italian opposition* → *Italian* (intersectivity) is not required, and blocking an inference of *Italian opposition* → *opposition* (non-restrictivity) is counterintuitive. The modifiers *last* and *former* are bound to NR, since *last/former president of the European Commission* does not entail *president of the European Commission*. The inflections of *be* are bound to IS, *of* is bound to OF and *a* to A. Similarly to the previous example, the label construction *The head of the Italian opposition Romano Prodi* is analyzed as an appositive. This is done by adding a token element, *APP* that is bound to WHO_A.

3. Expressing lexical relations: As mentioned above, this example requires expressing the lexical inference *last* → *former*. This is done by defining an entailment context as follows:
   $Ctx =$
   $\forall A_{et}.\forall x_e.last_{(et)et}(A)(x) \rightarrow former_{(et)et}(A)(x)$

4. Proof of Entailment:
   Let M be an intended model,
   ⟦[[The/THE [head/N_2 [of/OF [the/THE [Italian/MR opposition/N_1]]]]] APP/WHO_A [Romano Prodi]/NP], [was/IS$_{eq}$[the/THE [last/NR [president /N_2 [of/OF [the/THE [European Commission]/NP]]]]]⟧^{M}

$= (\mathit{was}/\textsc{is}_{eq}(\mathit{the}/\textsc{the}(\mathit{last}/\textsc{nr}(\mathit{president}/\textsc{n\_2}(\mathit{of}/\textsc{of}(\mathit{the}/\textsc{the}([\mathit{European\ Commission}]/\textsc{n\_1})))))))((\textsc{who\_a}([\mathit{Romano\ Prodi}]/\textsc{np}))(\mathit{the}/\textsc{the}(\mathit{head}/\textsc{n\_2}([\mathit{of}/\textsc{of}([\mathit{the}/\textsc{the}([\mathit{Italian}/\textsc{mr}(\mathit{opposition}/\textsc{n\_1})))))))$	analysis
$= (\mathit{was}/\textsc{is}_{eq}(\mathit{the}/\textsc{the}(\mathit{last}/\textsc{nr}(\mathit{president}/\textsc{n\_2}(\mathit{of}/\textsc{of}(\mathit{the}/\textsc{the}([\mathit{European\ Commission}]/\textsc{n\_1})))))))((\textsc{who\_a}(\lambda A_{et}.A(\mathbf{r}_e)))((\lambda A_{et}.\lambda B_{et}.B(\iota(A)))((\lambda A_{(et)t}.\lambda x_e.\ A(\lambda y_e.\mathbf{head}_{eet}(y)(x)))((\lambda A_{(et)t}.\lambda B_{((et)t)et}.\lambda x_e.\ B(A)(x))((\lambda A_{et}.\lambda B_{et}.B(\iota(A)))((\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{italian}_{(et)et}(A)(z))(\mathbf{opposition})))))))$	denotations of $\textsc{n\_1}$, $\textsc{n\_2}$, $\textsc{mr}$
$= (\mathit{was}/\textsc{is}_{eq}(\mathit{the}/\textsc{the}(\mathit{last}/\textsc{nr}(\mathit{president}/\textsc{n\_2}(\mathit{of}/\textsc{of}(\mathit{the}/\textsc{the}([\mathit{European\ Commission}]/\textsc{n\_1})))))))((\textsc{who\_a}(\lambda A_{et}.A(\mathbf{r}_e)))((\lambda B_{et}.B(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x))))))))$	func. app.
$= (\mathit{was}/\textsc{is}_{eq}(\mathit{the}/\textsc{the}(\mathit{last}/\textsc{nr}(\mathit{president}/\textsc{n\_2}(\mathit{of}/\textsc{of}(\mathit{the}/\textsc{the}([\mathit{European\ Commission}]/\textsc{n\_1})))))))(((\lambda A_{(et)t}.\lambda B_{(et)t}.\lambda C_{et}.C(\iota\ (\lambda x_e.B(=\ (x)) \wedge A(=(x)))))(\lambda A_{et}.A(\mathbf{r}_e)))((\lambda B_{et}.B(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x))))))))$	denotation of $\textsc{who\_a}$
$= (\mathit{was}/\textsc{is}_{eq}(\mathit{the}/\textsc{the}(\mathit{last}/\textsc{nr}(\mathit{president}/\textsc{n\_2}(\mathit{of}/\textsc{of}(\mathit{the}/\textsc{the}([\mathit{European\ Commission}]/\textsc{n\_1})))))))(((\lambda C_{et}.C(\iota\ (\lambda x_e.\ =\ (x)(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x)))))\wedge = (x)(\mathbf{r}_e))))))$	func. app.
$= (\mathit{was}/\textsc{is}_{eq}((\lambda A_{et}.\lambda B_{et}.B(\iota(A)))((\lambda A_{et}.\lambda x_e.(\mathbf{last}_{(et)et}(A))(x))((\lambda A_{(et)t}.\lambda x_e.\ A(\lambda y_e.\mathbf{president}_{eet}(y)(x)))((\lambda A_{(et)t}.\lambda B_{((et)t)et}.\lambda x_e.\ B(A)(x))((\lambda A_{et}.\lambda B_{et}.B(\iota(A)))(\mathbf{ec}_{et}))))))))\ (((\lambda C_{et}.C(\iota(\lambda x_e. = (x)(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\ \mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))\ (x)))))\wedge\ = (x)(\mathbf{r}_e))))))$	$\textsc{n\_1}$, $\textsc{n\_2}$, $\textsc{nr}$
$= (\mathit{was}/\textsc{is}_{eq}((\lambda A_{et}.\lambda B_{et}.B(\iota(A)))(\lambda x_e.\ (\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}\ (\iota(\mathbf{ec}_{et}))(y)))(x))))\ (((\lambda C_{et}.C(\iota(\lambda x_e.\ =\ (x)(\iota(\lambda x_e.\ ((\mathbf{head}_{eet}(\iota(\lambda y_e.\ \mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et}\ (\mathbf{opposition}))(y)))\ (x)))))\wedge\ = (x)(\mathbf{r}_e))))))$	func. app.

$= ((\lambda A_{(et)t}.\lambda B_{(et)t}.\ A(\lambda x_e.B(\lambda y_e.\ =\ (x)(y)))) ((\lambda B_{et}.B(\iota(\lambda x_e.\ (\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet} (\iota(\mathbf{ec}_{et}))(y)))(x)))))) (((\lambda C_{et}.C(\iota(\lambda x_e. = (x)(\iota(\lambda x_e. ((\mathbf{head}_{eet}(\iota(\lambda y_e.\ \mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et} (\mathbf{opposition}))(y)))\ (x)))))\wedge = (x)(\mathbf{r}_e))))))$	denotation of $\textsc{is}_{eq}$, func. app.
$= (\iota(\lambda x_e.(\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}(\iota(\mathbf{ec}_{et}))(y))) (x)))(\iota(\lambda x_e. = (x)(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition} (y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x)))))\wedge = (x)(\mathbf{r}_e))) = (*))$	func app.

In this example we have five applications of the iota operator that correspond to five definite descriptions:

(a) $\iota(\mathbf{ec}_{et})$
(*the European Commission*)

(b) $\iota(\lambda x_e.(\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}(\iota(\mathbf{ec}_{et}))(y)))(x))$
(*the last president of the European Commission*)

(c) $\iota(\lambda y_e.\mathbf{opposition}\ (y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y))$
(*the Italian opposition*)

(d) $\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}\ (y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x))))$
(*the head of the Italian opposition*)

(e) $\iota(\lambda x_e. = (x)(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}\ (y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x)))))\wedge = (x)(\mathbf{r}_e))$
(*the entity who is both the head of the Italian opposition and Romano Prodi*)

We can now add the corresponding uniqueness requirements to the entailment context:
$Ctx =$
$[\forall A_{et}.\forall x_e.last_{(et)et}(A)(x) \rightarrow former_{(et)et}(A)(x)] \wedge$

$[\forall_{(et)t}(\lambda a_e.\mathbf{ec}(a) \wedge \forall_{(et)t}(\lambda b_e.\mathbf{ec}(b) \rightarrow= (a)(b)))] \wedge$

$[\forall_{(et)t}(\lambda a_e.(\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}(\iota(\mathbf{ec}_{et}))(y)))(a) \wedge$
$\forall_{(et)t}(\lambda b_e.(\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}(\iota(\mathbf{ec}_{et}))(y)))(b) \rightarrow= (a)(b)))] \wedge$

$[\forall_{(et)t}(\lambda a_e.(\mathbf{opposition}(a) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(a)) \wedge \forall_{(et)t}$
$(\lambda b_e.(\mathbf{opposition}(b) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(b)) \rightarrow= (a)(b)))] \wedge$

$[\forall_{(et)t}(\lambda a_e.\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}(y) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))$
$(y)))(a) \wedge \forall_{(et)t}(\lambda b_e.\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}\ (y) \wedge (\mathbf{italian}_{(et)et}$
$(\mathbf{opposition}))(y)))(b) \rightarrow= (a)(b)))] \wedge$

$[\forall_{(et)t}(\lambda a_e. = (a)(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}\ (y)\wedge$
$(\mathbf{italian}_{(et)et}\ (\mathbf{opposition}))(y)))(x)))))\wedge = (a)(\mathbf{r}_e)\wedge$
$\forall_{(et)t}(\lambda b_e. = (b)(\iota(\lambda x_e.((\mathbf{head}_{eet}(\iota(\lambda y_e.\mathbf{opposition}\ (y)\wedge$
$(\mathbf{italian}_{(et)et}(\mathbf{opposition}))(y)))(x)))))\wedge = (b)(\mathbf{r}_e) \rightarrow= (a)(b)))]$

The existence requirements of all descriptions are given by the formula:
$\exists_{(et)t}(\lambda a_e.\mathbf{ec}_{et}(a) \wedge \exists_{(et)t}(\lambda b_e.((\mathbf{last}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}(a)(y)))(b)) \wedge$
$\exists_{(et)t}(\lambda c_e.\mathbf{opposition}(c) \wedge (\mathbf{italian}_{(et)et}(\mathbf{opposition}))(c) \wedge \exists_{(et)t}(\lambda d_e.$
$\mathbf{head}_{eet}(c)(d) \wedge \exists_{(et)t}(\lambda e_e. = (e)(d)\wedge = (e)(\mathbf{r}_e))))))$

We can now continue the proof from (*):

$$\begin{aligned}
&= Ctx\ \wedge\exists_{(et)t}(\lambda a_e.\mathbf{ec}_{et}(a) \wedge \exists_{(et)t}(\lambda b_e.((\mathbf{last}_{(et)et}(\lambda y_e. && \text{treatment}\\
&\quad \mathbf{president}_{eet}(a)(y)))(b)) \wedge \exists_{(et)t}(\lambda c_e.\mathbf{opposition}(c) && \text{of iotas}\\
&\quad \wedge(\mathbf{italian}_{(et)et}(\mathbf{opposition}))(c) \wedge \exists_{(et)t}(\lambda d_e. &&\\
&\quad \mathbf{head}_{eet}(c)(d) \wedge \exists_{(et)t}(\lambda e_e. = (e)(d)\wedge = (e)(\mathbf{r}_e) &&\\
&\quad \wedge = (b)(e)))))) &&\\
&\leq Ctx\ \wedge\exists_{(et)t}(\lambda a_e.\mathbf{ec}_{et}(a) \wedge \exists_{(et)t}(\lambda b_e.((\mathbf{last}_{(et)et}(\lambda y_e. && \text{def. of } \wedge\\
&\quad \mathbf{president}_{eet}(a)(y)))(b))\wedge = (b)(\mathbf{r}_e))) &&\\
&\leq \exists_{(et)t}(\lambda a_e.\mathbf{ec}_{et}(a) \wedge \exists_{(et)t}(\lambda b_e.((\mathbf{former}_{(et)et}(\lambda y_e. && \text{lexical in-}\\
&\quad \mathbf{president}_{eet}(a)(y)))(b))\wedge = (b)(\mathbf{r}_e))) && \text{ference}\\
&= \exists_{(et)t}(\lambda a_e.\mathbf{ec}_{et}(a)\ \wedge\ ((\mathbf{former}_{(et)et}(\lambda y_e. &&\\
&\quad \mathbf{president}_{eet}(a)(y)))(\mathbf{r}_e))) &&\\
&= ((\mathbf{former}_{(et)et}(\lambda y_e.\mathbf{president}_{eet}(\iota(\lambda a_e.\mathbf{ec}_{et}(a)))(y))) && \text{treatment}\\
&\quad (\mathbf{r}_e)) && \text{of iota}
\end{aligned}$$

$= [\![$[Romano Prodi]/NP [is/IS [a/A [former/NR [president/N_2 [of/OF
[the/THE [European Commission]/N_1]]]]]]$]\!]^M$

To summarize, this example illustrates the use of the formal model to account for entailments that include lexical inferences. Such inferences are expressed in lambda calculus formula(s) as part of an intermediate stage in the analysis of a text-hypothesis pair, and serve as assumptions posited for proving the entailment.

4.4.3 More on Restrictivity and Intersectivity

Let us use the following hypothetical example to investigate further advantages of the distinction between restrictive and intersective modifiers, as modeled in the lexicon.

1. Pair 1

   t: Jan is a short Dutch man.

   h: Jan is a short man.

   This pair does not involve an intuitive entailment: a short Dutch man may be tall by world standards.

2. Pair 2

   t: Jan is a black Dutch man.

   h: Jan is a black man.

   This pair involves an intuitive entailment: a black Dutch man is a man who is both black and Dutch, hence he is a black man.

From a purely syntactic standpoint, these two *t*–*h* pairs are indistinguishable. In both pairs, the lexical overlap between t and h is 100% and the phrase structures are identical. Furthermore, syntactically, both *short* and *black* are adjectives attached to the noun phrase *Dutch man*. This syntactic configuration contains no indication that omitting *Dutch* in Pair 1 might result in a different entailment classification than doing the same in Pair 2. Semantically, however, both the non-validity the entailment in Pair 1 and the validity the entailment in Pair 2 can be analyzed based on the modeling restrictivity and intersectivity in the interpreted lexicon.

Analysis of Pair 1

To validate the absence of entailment between a text and a hypothesis is to show that there is an intended model $M = \langle E, I \rangle$ in which there is no $\leq$ relation between their denotations.

Let us posit parse trees and bindings as follows:

t: Jan/NP [is/IS [a/A [short/MR [Dutch/MI man/N_1]]]]

h: Jan/NP [is/IS [a/A [short/MR man/N_1]]]

Let M be an intended model that satisfies the following:

- $\mathbf{man}_{et}$ characterizes $\{\mathbf{dan}, \mathbf{jan}, \mathbf{vim}\}$
- $\mathbf{dutch}_{et}$ characterizes $\{\mathbf{jan}, \mathbf{vim}\}$
- $\mathbf{short}(\mathbf{man})_{et}$ characterizes $\{\mathbf{dan}\}$
- $\mathbf{short}(\lambda y_e.\mathbf{man}(y) \wedge \mathbf{dutch}(y))_{et}$ characterizes $\{\mathbf{jan}\}$

Consider the denotations of t and h in M:

- Text:

  $[\![Jan/\textsc{np}\ [is/\textsc{is}\ [a/\textsc{a}\ [short/\textsc{mr}\ [Dutch/\textsc{mi}\ man/\textsc{n\_1}]]]]]\!]^M$

$= (is/\textsc{is}(a/\textsc{a}(short/\textsc{mr}(Dutch/\textsc{mi}(man/\textsc{n\_1})))))(Jan/\textsc{np})$	analysis
$= (is/\textsc{is}(a/\textsc{a}((\lambda A_{et}.\lambda x_e.A(x) \wedge (\mathbf{short}_{(et)et}(A))(x))$ $((\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{dutch}_{et}(x))(\mathbf{man}_{et})))))(Jan/\textsc{np})$	denotations of MR, MI, N_1
$= (is/\textsc{is}(a/\textsc{a}((\lambda A_{et}.\lambda x_e.A(x) \wedge (\mathbf{short}_{(et)et}(A))$ $(x))(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x)))))(Jan/\textsc{np})$	func. app.
$= (is/\textsc{is}(a/\textsc{a}(\lambda x_e.(\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x)\wedge$ $(\mathbf{short}_{(et)et}(\lambda y_e.\mathbf{man}_{et}(y) \wedge \mathbf{dutch}_{et}(y)))(x)))))$ $(Jan/\textsc{np})$	func. app.
$= ((\lambda A_{(et)t}.\lambda B_{(et)t}.\ A(\lambda x_e.B(\lambda y_e. = (x)(y))))$ $((\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.A(x) \wedge B(x)))(\lambda x_e.(\mathbf{man}_{et}(x)\wedge$ $\mathbf{dutch}_{et}(x)\wedge (\mathbf{short}_{(et)et}(\lambda y_e.\mathbf{man}_{et}(y)\wedge\mathbf{dutch}_{et}(y)))$ $(x)))))(\lambda A_{et}.A(\mathbf{jan}_e))$	denotations of IS, A, NP
$= ((\lambda A_{(et)t}.\lambda B_{(et)t}.\ A(\lambda x_e.B(\lambda y_e. = (x)(y))))$ $(\lambda C_{et}.\exists_{(et)t}\ (\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge$ $(\mathbf{short}_{(et)et}(\lambda y_e.\mathbf{man}_{et}(y) \wedge \mathbf{dutch}_{et}(y)))(x) \wedge$ $C(x))))(\lambda A_{et}.A(\mathbf{jan}_e))$	func. app.
$= (\lambda B_{(et)t}.(\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x)\wedge$ $(\mathbf{short}_{(et)et}(\lambda y_e.\mathbf{man}_{et}(y) \wedge \mathbf{dutch}_{et}(y)))(x) \wedge C(x)))$ $(\lambda v_e.B(\lambda z_e. = (v)(z))))(\lambda A_{et}.A(\mathbf{jan}_e))$	func. app.
$= (\lambda B_{(et)t}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}\ (x)\wedge$ $(\mathbf{short}_{(et)et}\ (\lambda y_e.\mathbf{man}_{et}(y) \wedge \mathbf{dutch}_{et}(y)))(x) \wedge$ $B(\lambda z_e. = (x)(z))))\ (\lambda A_{et}.A(\mathbf{jan}_e))$	func. app.
$= \exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge (\mathbf{short}_{(et)et}$ $(\lambda y_e.\mathbf{man}_{et}(y) \wedge \mathbf{dutch}_{et}(y)))(x)\wedge = (x)(\mathbf{jan}_e))$	func. app.
$= \mathbf{man}_{et}(\mathbf{jan}_e) \wedge \mathbf{dutch}_{et}(\mathbf{jan}_e) \wedge (\mathbf{short}_{(et)et}$ $(\lambda y_e.\mathbf{man}_{et}(y) \wedge \mathbf{dutch}_{et}(y)))(\mathbf{jan}_e)$	func. app.
$= 1 \wedge 1 \wedge 1$	denotations in M
$= 1$	def of $\wedge$

- Hypothesis:

  $[\![\mathit{Jan}/\textit{NP}\ [\mathit{is}/\textit{IS}\ [\mathit{a}/\textit{A}\ [\mathit{short}/\textit{MR}\ \mathit{man}/\textit{N\_1}]]]]]\!]^M$

$$
\begin{aligned}
&= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}(\mathit{short}/\textsc{mr}(\mathit{man}/\textsc{n\_1}))))(\mathit{Jan}/\textsc{np}) && \text{analysis}\\
&= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}((\lambda A_{et}.\lambda x_e.A(x) \wedge (\mathbf{short}_{(et)et}(A))(x)) && \text{denotations}\\
&\quad (\mathbf{man}_{et}))))(\mathit{Jan}/\textsc{np}) && \text{of } \textsc{mr}\text{, } \textsc{n\_1}\\
&= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}(\lambda x_e.\mathbf{man}_{et}(x) \wedge (\mathbf{short}_{(et)et}(&& \text{func. app.}\\
&\quad \mathbf{man}_{et}))(x))))(\mathit{Jan}/\textsc{np}) && \\
&= ((\lambda A_{(et)t}.\lambda B_{(et)t}.A(\lambda x_e.B(\lambda y_e. = (x)(y)))) && \text{denotations}\\
&\quad ((\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.A(x) \wedge B(x)))\ (\lambda x_e.\mathbf{man}_{et}(x) \wedge && \text{of } \textsc{is}\text{, } \textsc{a}\text{, } \textsc{np}\\
&\quad (\mathbf{short}_{(et)et}(\mathbf{man}_{et}))(x))))\ (\lambda A_{et}.A(\mathbf{jan}_e)) && \\
&= ((\lambda A_{(et)t}.\lambda B_{(et)t}.A(\lambda x_e.B(\lambda y_e. = (x)(y)))) && \text{func. app.}\\
&\quad (\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge (\mathbf{short}_{(et)et}\ (\mathbf{man}_{et}))(x) \wedge && \\
&\quad C(x))))\ (\lambda A_{et}.A(\mathbf{jan}_e)) && \\
&= (\lambda B_{(et)t}.(\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge && \text{func. app.}\\
&\quad (\mathbf{short}_{(et)et}(\mathbf{man}_{et}))(x) \wedge C(x)))\ (\lambda x_e.B(\lambda y_e. = && \\
&\quad (x)(y))))(\lambda A_{et}.A(\mathbf{jan}_e)) && \\
&= (\lambda B_{(et)t}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge (\mathbf{short}_{(et)et} && \text{func. app.}\\
&\quad (\mathbf{man}_{et}))(x) \wedge B(\lambda y_e. = (x)(y))))(\lambda A_{et}.A(\mathbf{jan}_e)) && \\
&= \exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge (\mathbf{short}_{(et)et}(\mathbf{man}_{et}))\ (x) \wedge = && \text{func. app.}\\
&\quad (x)(\mathbf{jan}_e)) && \\
&= \mathbf{man}_{et}(\mathbf{jan}_e) \wedge (\mathbf{short}_{(et)et}\ (\mathbf{man}_{et}))(\mathbf{jan}_e) && \text{func. app.}\\
&= 1 \wedge 0 && \text{denotations}\\
& && \text{in } M\\
&= 0 && \text{def. of } \wedge
\end{aligned}
$$

Intuitively, *Jan* can be a man who is considered to be short in the population of Dutch men, hence $(\mathbf{short}(\lambda x_e.\mathbf{man}(x) \wedge \mathbf{dutch}(x)))\ (\mathbf{jan})$ would return 1; but not in the population of all men, hence $(\mathbf{short}\ (\mathbf{man}))(\mathbf{jan})$ would return 0. This is a direct consequence of *short* denoting a non-intersective modifier: the set denoted by $\mathbf{short}(\lambda x_e.\mathbf{man}(x) \wedge \mathbf{dutch}(x))$ is not necessarily a subset of $\mathbf{short}(\mathbf{man})$.

Analysis of Pair 2

Let us posit annotated parse trees as follows:

t: Jan/NP [is/IS [a/A [black/MI [Dutch/MI man/N_1]]]]

h: Jan/NP [is/IS [a/A [black/MI man/N_1]]]

A proof of entailment:

Let M be an intended model,

$[\![\mathit{Jan}/\textit{\textsc{np}}\ [\mathit{is}/\textit{\textsc{is}}\ [\mathit{a}/\textit{\textsc{a}}\ [\mathit{black}/\textit{\textsc{mi}}\ [\mathit{Dutch}/\textit{\textsc{mi}}\ \mathit{man}/\textit{\textsc{n\_1}}]]]]]\!]^M$

$= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}(\mathit{black}/\textsc{mi}(\mathit{Dutch}/\textsc{mi}(\mathit{man}/\textsc{n\_1})))))(\mathit{Jan}/\textsc{np})$	analysis
$= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}((\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{black}_{et}(x))((\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{dutch}_{et}(x))(\mathbf{man}_{et})))))(\mathit{Jan}/\textsc{np})$	denotations of MI, N_1
$= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}((\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{black}_{et}(x))(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x)))))(\mathit{Jan}/\textsc{np})$	func. app.
$= (\mathit{is}/\textsc{is}(\mathit{a}/\textsc{a}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge \mathbf{black}_{et}(x))))(\mathit{Jan}/\textsc{np})$	func. app.
$= ((\lambda A_{(et)t}.\lambda B_{(et)t}.A(\lambda x_e.B(\lambda y_e.\ =\ (x)(y))))((\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.A(x) \wedge B(x)))\ (\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge \mathbf{black}_{et}(x))))(\lambda A_{et}.A(\mathbf{jan}_e))$	denotations of IS, A, NP
$= ((\lambda A_{(et)t}.\lambda B_{(et)t}.A(\lambda x_e.B(\lambda y_e.\ =\ (x)(y))))(\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge \mathbf{black}_{et}(x))))(\lambda A_{et}.A(\mathbf{jan}_e))$	func. app.
$= (\lambda B_{(et)t}.(\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}\ (x) \wedge \mathbf{black}_{et}(x)))(\lambda v_e.B(\lambda z_e. = (v)(z))))(\lambda A_{et}.\ A(\mathbf{jan}_e))$	func. app.
$= (\lambda B_{(et)t}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge \mathbf{black}_{et}(x) \wedge B(\lambda z_e. = (x)(z))))(\lambda A_{et}.\ A(\mathbf{jan}_e))$	func. app.
$= \exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{dutch}_{et}(x) \wedge \mathbf{black}_{et}(x) \wedge\ = (x)(\mathbf{jan}_e))$	func. app.
$= \mathbf{man}_{et}(\mathbf{jan}_e) \wedge \mathbf{dutch}_{et}(\mathbf{jan}_e) \wedge \mathbf{black}_{et}(\mathbf{jan}_e)$	func. app.
$\leq \mathbf{man}_{et}(\mathbf{jan}_e) \wedge \mathbf{black}_{et}(\mathbf{jan}_e)$	def. of ∧
$= \exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{black}_{et}(x) \wedge = (x)(\mathbf{jan}_e))$	func. app.
$= (\lambda B_{(et)t}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{black}_{et}(x) \wedge B(\lambda z_e. = (x)(z))))(\lambda A_{et}.\ A(\mathbf{jan}_e))$	func. app.
$= (\lambda B_{(et)t}.(\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{black}_{et}(x)))(\lambda v_e.B(\lambda z_e. = (v)(z))))(\lambda A_{et}.A(\mathbf{jan}_e))$	func. app.
$= ((\lambda A_{(et)t}.\lambda B_{(et)t}.A(\lambda x_e.B(\lambda y_e.\ =\ (x)(y))))(\lambda C_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{black}_{et}(x))))(\lambda A_{et}.A(\mathbf{jan}_e))$	func. app.

$$\begin{array}{ll|l}
= & ((\lambda A_{(et)t}.\lambda B_{(et)t}.A(\lambda x_e.B(\lambda y_e. = (x)(y)))) & \text{func. app.} \\
 & ((\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.A(x) \wedge B(x)))\ (\lambda x_e.\mathbf{man}_{et}(x) \wedge & \\
 & \mathbf{black}_{et}(x))))(\lambda A_{et}.A(\mathbf{jan}_e)) & \\
= & (is/\textsc{is}(a/\textsc{a}(\lambda x_e.\mathbf{man}_{et}(x) \wedge \mathbf{black}_{et}(x))))(Jan/\textsc{np}) & \text{denotations} \\
 & & \text{of } \textsc{is},\ \textsc{a},\ \textsc{np} \\
= & (is/\textsc{is}(a/\textsc{a}((\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{black}_{et}(x))(\mathbf{man}_{et})))) & \text{func. app.} \\
 & (Jan/\textsc{np}) & \\
= & (is/\textsc{is}(a/\textsc{a}(black/\textsc{mi}(man/\textsc{n\_1}))))(Jan/\textsc{np}) & \text{denotations} \\
 & & \text{of } \textsc{mi},\ \textsc{n\_1} \\
= & [\![\, Jan\ [is\ [a\ [black\ man]]] \,]\!]^M & \text{analysis}
\end{array}$$

In this case the transition relies on the intersectivity of *black*, which in conjunction with the intersectivity of *Dutch* licenses the inference that the set characterized by the et function $[\![\, black\ [Dutch\ man] \,]\!]^M$ equals to the set characterized by $[\![\, Dutch\ [black\ man] \,]\!]^M$, which is a subset of the set characterized by $[\![\, black\ man \,]\!]^M$.

To summarize, based on semantic information that differentiates between the restrictive modifier *short* and the intersective modifiers *Dutch* and *black*, it is possible to draw a distinction between the non-entailment Pair 1 and the entailment Pair 2. Although this simple example was constructed here for illustrative purposes, the phenomena of restrictivity and intersectivity that it illustrates are prevalent in natural entailment data.

CHAPTER 5

Proof-Based Platform for Annotating Entailment

This chapter extends and completes the description of a proof-based annotation platform first appeared in Toledo et al. (2014b) and Toledo et al. (2014a).

Overview

This chapter presents an annotation platform for creating a corpus of entailment pairs in which the linguistic phenomena annotated allow a formal system to validate the inferences. The platform implements the semantic model of entailment presented in the previous chapter. A semi-automatic workflow is employed, in which: (a) sentences are parsed into tree structures and heuristically annotated based on part of speech tags, (b) human annotators correct parsing errors and verify that the annotations marked are according to the subjective inferential process by which they understand the entailment, and (c) a proof system calculates semantic terms for the text and hypothesis based on the annotations and validates that the term of the hypothesis is a logical consequence of the term of the text. The platform was the basis for the creation of the annotated corpus SemAnTE 2.0 described in Chapter 6.

5.1 Introduction

In the last chapter we have seen how a standard formal semantic theory is capable of providing theoretical explanations for entailments in natural language.

The central component in this framework is the interpreted lexicon, which associates lexical items with abstract representations. Such representations are shown to be sufficiently expressive for capturing natural inferences.

Our goal in this chapter is to show an implementation of the semantic model in a computational framework. In addition to that, we aim to automate the analysis to a large extent in order to reduce the effort required from human annotators. This was made possible because the formal approach lends itself to an automatic verification that the text entails the hypothesis based on the marked annotations. This verification is achieved by employing a theorem prover to find a formal deduction that the analysis of the hypothesis is a logical consequence of the analysis of the text. Thus, the annotation platform can be treated as a proof system.

A basic requirement from a proof system is to be logically sound with respect to a semantics such as the formal model described in Chapter 4. Soundness guarantees that any hypothesis that the proof system deduces from a text is a semantically correct inference in the sense of satisfying the truth-conditionality criterion defined in Section 4.2. However, the formal model of Chapter 4 is more expressive than the first-order methods in our proof system implementation. Thus, in principle, it is possible to encounter a valid entailment that cannot be proven by the system.[1]

The proof system of SemAnTE is designed to work on a pair of text and hypothesis that are parsed into phrase structures and annotated with lexical items of linguistic phenomena from the lexicon. The parsing and the annotation tasks are performed in a semi-automatic manner: the parsing is done using a standard stochastic parser and subsequently, the annotations are marked using a rule-based heuristic that relies on part of speech tags. The output of these processes is reviewed by human annotators, corrected and completed manually if needed. A graphical user interface is used for this manual work. The system calculates semantic terms for the text and hypothesis and validates formally the inference from the term of the text and that of the hypothesis. This platform was used for the creation of the corpus SemAnTE 2.0 described in Chapter 6.

This chapter has two parts. The first one focuses on the architecture of the annotation platform and describes its internal components. The second one explains how the system was used in practice and illustrates the workflow with an example.

5.2 Components of the Platform

The annotation platform is made up of several components that are responsible for processing the text and hypothesis in a sequential order.

Figure 5.1 shows the components and the dataflow in their actual use. The first components are responsible for parsing sentences into phrase structures

[1] In practice no such case has been encountered.

and for binding their words to the lexicon automatically based on an annotation heuristic. The rest of the components make up the remaining parts in the proof system processing: (1) – a type consistency checker which verifies that the provided structure and annotations satisfy the requirements of the formal model, (2) – a beta-reducer that performs function applications to simplify the terms of the text and hypothesis, (3) – an order lowering component that is responsible for reducing the terms from higher-order logic to first order, and (4) – a theorem prover is used for checking whether the term generated for the hypothesis is deducible from the term generated for the text.

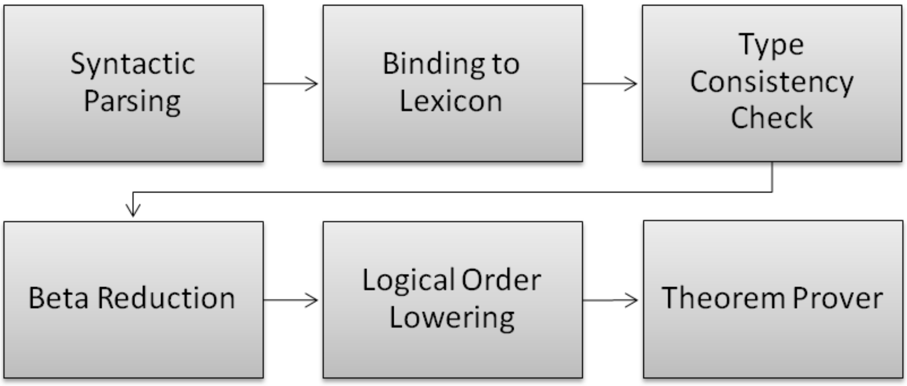

Figure 5.1: Components and Dataflow

5.2.1 Syntactic Parsing

This component generates phrase structures annotated with part of speech tags for the text and hypothesis. The phrase structures specify the function-argument relations between words and phrases, thus allowing the correct order in the composition of terms within the sentence analysis. Part of speech tags are useful for binding words to the lexicon as explained in Section 5.2.2.

To illustrate the contribution of a phrase structure to the analysis of a sentence, consider the following example with annotations: *John/NP ate/V_2 an/A apple/N_1*. The V_2 annotation of the verb *eat* specifies that it has two argument positions. However, without a tree structure, we under-specify the order in which the verb takes its arguments. Thus we have an ambiguity between two analyses: the correct one, in which John is the agent of the eating and the apple is the patient, and an incorrect one in which John is the patient of the eating and the apple is the agent.

A phrase structure analysis licenses only one interpretation since it specifies the order in which words compose into phrases, phrases into bigger phrases, and eventually into sentences. This information derives an unambiguous structure and therefore allows us to derive only the correct meaning.

The platform uses the robust stochastic parser of Klein and Manning (2003) and also employs the part of speech tagger of Toutanova et al. (2003). These are executed as part of the NLP package of Manning et al. (2014).[2]

Manual work is required in order to guarantee that the parse trees obtained from the parser are structured correctly. This work is done by human annotators in a graphical user interface (UI).

The main reason for correcting parse trees is parsing errors due to the imperfection of statistical parsers (Type 1). The secondary reason is differences between the model of syntax that the parser was trained on and the one adopted in the semantic model (Type 2). To illustrate, consider the text of Example (1) repeated below in (41a). The parse tree generated by the parser is shown in (41b) and the tree after correction appears in (41c).

(41) a. The largest search engine on the web, Google, receives over 200 million queries each day through its various services.

b. [[The [largest [search engine]]] [on [the web, Google]]], [[receives [over [[200 million] queries] [each day]]] [through [its [various services]]]]

c. [[The [largest [[search engine] [on the web]]]], Google], [[receives [[over [200 million]] queries] [each day]] [through [its various services]]]

A Type 1 error appeared in the analysis of the the appositive construction, originally parsed as *[[The [largest [search engine]]] [on [the web, Google]]]*. This was corrected to *[[The [largest [[search engine] [on the web]]]], Google]*. A Type 2 error appeared in the attachment of a modifier, *on the web, Google*, to a full noun phrase, *the largest search engine*. This was corrected by attaching the modifier to the internal noun compound, *search engine*. The parser employed in the platform systematically attaches modifiers of noun phrases to the whole phrase rather than the internal noun. This is a known theoretical issue in the formal semantic literature that goes back to Partee (1975). In this work, Partee motivates the analysis adopted here, where the modifier is attached to the noun rather than the whole noun phrase, based on the interpretation of definite descriptions.[3]

[2] Version 1.3.4

[3] In a nutshell, *The student got an A in the exam* presupposes that there is only one student in the context of utterance, but *The student of Linguistics got an A in the exam* presupposes the existence of a single student of linguistics. There can be several students in the context, but one and only one student of linguistics. This indicates that the definite article is attached to the complex noun *student of linguistics*. Thus the analysis is *The [student [of Linguistics]]* rather than *[The student] [of Linguistics]*.

5.2.2 Binding to Lexicon

This component is responsible for marking the linguistic phenomena that allows a formal analysis of the inference between *t* and *h*. This is done by binding the words and phrases of *t* and *h* to lexical items from the lexicon. The annotation is done over the parse trees obtained by the previous component.

The annotation process is based on a heuristic part, which is done automatically based on part-of-speech tag, and is later corrected by the human This heuristic is implemented as a set of annotation rules. The rules are summarized in Table 5.1. For example, when a token has the part of speech of *VB* and its text is an inflection of *Be*, such as *am*, *is*, *are*, *was*, etc., it is annotated as IS.

Targeted Category	Token PoS	Token Text	Annotation
Conjunction	CC		AND
Indefinite nouns	NN		N
Definite nouns	NNP		NP
Inflections of *be*	VB	am/is/was/…	IS
Verbs	VB		V_1
Adjectives	JJ		MR
Adverbs	RB		MR
Prepositions	IN, TO		P_R
WH-words	WP, WDT		WHO_R
Indefinite articles		a/an	A
Definite article		the	THE
Possessive ’s		’s	GEN
Punctuation		(/)/,/.	IGNORE
Existential Quantifier		some	SOME
Universal Quantifier		every	EVERY

Table 5.1: Annotation Rules

The annotation rules aim to provide a starting point for manual work by the human annotators. In many examples this starting point is fairly close to the correct annotations but some manual corrections are usually needed. This is due to the over-simplicity of the heuristic.

For example, notice that all adjectives are annotated as restrictive modifiers (MR). The annotation of an intersective modifier (MI) is never employed by these rules. This is because the classification of an adjectival modifier as either restrictive or intersective is a complicated issue that is subject to world knowledge and contextual information. See Section 4.4.1 for discussion. These corrections are left for the human annotators. In addition, the heuristic binds all verbs to V_1. In case of a transitive verb an annotator overrides this annotation with V_2. Annotators review and fix the parse trees in almost every sentence, hence the extra effort in correcting the annotations is minor.[4]

Annotators may also need to insert new elements into the tree in order to address covert determiners, quantifiers or syntactic operators (as illustrated in Section 4.4.1).

In addition to that, manual work is required for indicating lexical relations. An example was given in Section 4.4.2. In such cases it is required to identify lexical inferences in the entailment and to indicate them as lexical relations that pair words or phrases in t with corresponding ones in h.

5.2.3 Type Constituency Check

This component determines whether the formal process of proof searching is at all possible for the given annotated trees of the text and hypothesis. It is a sanity check for the compositionality of the annotated trees.

As explained in Chapter 4, every entry in the interpreted lexicon is assigned a type by the typing function TYP. Therefore, the type of each word from t and h can be determined by the lexical item that the word is bound to. Based on the binary phrase structures in which the data is organized, the platform performs standard type-checking. This is done by checking that in each branching node one type is a function of type $\tau\sigma$ and the other is of type τ.

The type checking function, *type-check*, is defined according to the pseudocode in Figure 5.2. It is assumed that its argument, *element*, is either a leaf or a branching node in the tree. The function *branching-node*(element) returns $true$ if the given element is a branching node, and $false$ if it is a leaf. Furthermore, it is assumed that any leaf element is bound to a label defined in the lexicon and that branching nodes have a *left* and *right* sub-elements. A function for calculating the type of a branching node, *type-of* is also defined and calls the function TYP to retrieve the type of a label based on its definition in the lexicon. Note that all denotations in the lexicon have a pre-defined type, as shown in Tables 4.3 and 4.4. Therefore, the function TYP is well-defined for any label in the lexicon, including labels of template-based denotations.

Passing the type consistency check guarantees that function application is possible at any branching node in the tree. Hence the sentential terms of t and h can be calculated. Failing the check means that there is an annotation error that needs to be corrected by a human annotator. In that case the annotator fixes the error and restarts the type checking.

[4] The heuristic can be improved in the future to require less manual correction work.

```
function type-check(element)
    if type-of(element) is not undefined
        return success
    else
        return failure
end

function type-of(element)
    if branching-node(element)
        if (type-of(element.left)=$\tau\sigma$ $\wedge$ type-of(element.right)=$\tau$) $\vee$
          (type-of(element.left)=$\tau$ $\wedge$ type-of(element.right)=$\tau\sigma$)
            return $\sigma$
        else
            return undefined
    else
        return TYP(element.label)
end
```

Figure 5.2: Pseudo-code of the *type-check* Function

5.2.4 Beta Reduction

The platform uses a typed-lambda calculus engine to calculate terms that represent the sentential meanings of t and h based on their annotated parse trees. This operation is performed on t and h independently and has two steps: (a) transforming a tree to a term, and (b) simplifying the term using a rule of beta-reduction. Step (a) was implemented by Kokke (2013b) and step (b) by Kokke (2013a). Pseudo-code of these two steps appears in Figure 5.3, in the functions *tree-to-formula* and *beta-reduce*, respectively.

The rule of beta reduction simplifies instances of function application where a lambda abstraction $\lambda x_\tau . f_{\tau\sigma}(x_\tau)$ is applied to some argument y_τ. The rule returns the term $f_{\tau\sigma}[x_\tau := \mathit{rename}(f_{\tau\sigma}, y_\tau)]$ in which the occurrences of x in f are substituted by a copy of y created by the function *rename*. The renaming function creates fresh names for free variables in y in order to avoid accidental binding of these variables in f as a result of the substitutions.[5] The rule of beta reduction is checked iteratively until no more reductions are possible.

The pseudo-code in Figure 5.3 employs the following functions in addition to *branching-node*, mentioned in Section 5.2.3:

- *function-application*(term) returns *true* if the given term is a function application. The format of a function application is $f(x)$. Otherwise it returns $false$.

[5] This is sometimes referred to as *capture-avoiding substitution.*

- *get-application-func*(term) returns the function part of a function application term: *get-application-func*($f(x)$) returns f.

- *get-application-arg*(term) returns the argument part of a function application term: *get-application-arg*($f(x)$) returns x.

- *lambda-abstraction*(term) returns *true* if the given term is a lambda abstraction. The format of a lambda abstraction is $\lambda x.f(x)$. Otherwise it returns $false$.

- *get-abstraction-body*(term) returns the body part of a lambda abstraction term: *get-abstraction-body*($\lambda x.f(x)$) $= f(x)$.

- *get-abstraction-var*(term) returns the variable part of a lambda abstraction term: *get-abstraction-var*($\lambda x.f(x)$) $= x$.

- *define-function-app*(func,arg) creates a function application term from terms of a function and an argument. For example, calling the function with the terms g and x as arguments returns $g(x)$.

- *define-lambda-abstraction*(var,func) creates a lambda abstraction term from a variable term and a function. For example, calling the function with the terms x and g as arguments returns $\lambda x.g$.

- *not-equal*(term1,term2) returns *true* if term1 and term2 are not literally equal. We ignore differences in names of bound variables (α-*equivalence*), thus $\lambda x.x$ is judged as equal to $\lambda y.y$.

For illustration, let us calculate the term at the root of the following annotated tree: *Mary*/NP [*loves*/V_2 [*a*/A [*boy*/N_1]]]. We begin by transforming the tree to a formula using the function *tree-to-formula*:

tree-to-formula(*Mary*/NP [*loves*/V_2 [*a*/A [*boy*/N_1]]])
= (*tree-to-formula*(*Mary*/NP))(*tree-to-formula*([*loves*/V_2 [*a*/A [*boy*/N_1]]]))
= *Mary*/NP(*tree-to-formula*([*loves*/V_2 [*a*/A [*boy*/N_1]]]))
= *Mary*/NP((*tree-to-formula*(*loves*/V_2))(*tree-to-formula*([*a*/A [*boy*/N_1]])))
= *Mary*/NP(*loves*/V_2(*tree-to-formula*([*a*/A [*boy*/N_1]])))
= *Mary*/NP(*loves*/V_2((*tree-to-formula*(*a*/A))(*tree-to-formula*(*boy*/N_1))))
= *Mary*/NP(*loves*/V_2(*a*/A(*boy*/N_1)))

Then we call the function *beta-reduce*:

$\textit{beta-reduce}(\textit{Mary}/\textsc{np}(\textit{loves}/\textsc{v\_2}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))))$

$\rightarrow \textit{reduce-step}(\textit{Mary}/\textsc{np}(\textit{loves}/\textsc{v\_2}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))))$	iteration 1
$= \textit{reduce-step}((\lambda A_{et}.A(\mathbf{mary}_e))(\textit{loves}/\textsc{v\_2}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))))$	def. of NP
$\leftarrow (\textit{loves}/\textsc{v\_2}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1})))(\mathbf{mary})$	reduction
$\rightarrow \textit{reduce-step}((\textit{loves}/\textsc{v\_2}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1})))(\mathbf{mary}))$	iteration 2
$\quad\rightarrow \textit{reduce-step}(\textit{loves}/\textsc{v\_2}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1})))$	decompose app.
$\quad= \textit{reduce-step}(((\lambda A_{(et)t}.\lambda x_e.A(\lambda y_e.\mathbf{loves}_{eet}(y)(x))))$ $(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1})))$	def. of V_2
$\quad\leftarrow \lambda x_e.(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))(\lambda y_e.\mathbf{loves}(y)(x))$	reduction
$\leftarrow (\lambda x_e.(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))(\lambda y_e.\mathbf{loves}(y)(x)))(\mathbf{mary})$	recompose app.
$\rightarrow \textit{reduce-step}((\lambda x_e.(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))(\lambda y_e.\mathbf{loves}(\mathrm{y})(\mathrm{x})))$ $(\mathbf{mary}))$	iteration 3
$\leftarrow ((\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))(\lambda y_e.\mathbf{loves}(\mathrm{y})(\mathbf{mary})))$	reduction
$\rightarrow \textit{reduce-step}((\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))(\lambda y_e.\mathbf{loves}(\mathrm{y})(\mathbf{mary})))$	iteration 4
$\quad\rightarrow \textit{reduce-step}(\textit{a}/\textsc{a}(\textit{boy}/\textsc{n\_1}))$	decompose app.
$\quad= \textit{reduce-step}((\lambda A_{et}.\lambda B_{et}.\exists_{(et)t}(\lambda x_e.A(x) \wedge B(x)))$ $(\mathbf{boy}_{et}))$	def. of A, N_1
$\quad\leftarrow \lambda B_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge B(x))$	reduction
$\leftarrow (\lambda B_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge B(x)))(\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))$	recompose app.
$\rightarrow \textit{reduce-step}((\lambda B_{et}.\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge B(x)))$ $(\lambda y_e.\mathbf{loves}(y)(\mathbf{mary})))$	iteration 5
$\leftarrow (\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge (\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))(x)))$	reduction
$\rightarrow \textit{reduce-step}((\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge (\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))$ $(x))))$	iteration 6
$\quad\rightarrow \textit{reduce-step}(\exists_{(et)t})$	decompose app.
$\quad\leftarrow \exists_{(et)t}$	return value
$\quad\rightarrow \textit{reduce-step}(\lambda x_e.\mathbf{boy}(x) \wedge (\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))(x))$	decompose app.
$\quad= \textit{reduce-step}(\lambda x_e.(\wedge\mathbf{boy}(x))(\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))$ $(x))$	$\wedge$ notation

$$\begin{array}{l|l}
\rightarrow \textit{reduce-step}((\wedge\mathbf{boy}(x))(\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))(x)) & \text{decompose abs.} \\
\quad\rightarrow \textit{reduce-step}(\wedge\mathbf{boy}(x)) & \text{decompose app.} \\
\quad\leftarrow \wedge\mathbf{boy}(x) & \text{return value} \\
\quad\rightarrow \textit{reduce-step}((\lambda y_e.\mathbf{loves}(y)(\mathbf{mary}))(x)) & \text{decompose app.} \\
\quad\leftarrow \mathbf{loves}(x)(\mathbf{mary}) & \text{return value} \\
\leftarrow (\wedge\mathbf{boy}(x))(\mathbf{loves}(x)(\mathbf{mary})) & \text{recompose app.} \\
\leftarrow \lambda x_e.(\wedge\mathbf{boy}(x))(\mathbf{loves}(x)(\mathbf{mary})) & \text{recompose abs.} \\
= \lambda x_e.\mathbf{boy}(x) \wedge \mathbf{loves}(x)(\mathbf{mary}) & \wedge\ \text{notation} \\
\leftarrow \exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge \mathbf{loves}(x)(\mathbf{mary})) & \text{recompose app.}
\end{array}$$

Hence, the term at the root of the tree *Mary*/NP [*loves*/V_2 [*a*/A [*boy*/N_1]]] is: $\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge \mathbf{loves}(x)(\mathbf{mary}))$.[6]

5.2.5 Handling of Iotas, Zetas and Lexical Relations

Instances of the ι operator are handled as explained in Section 4.3.5. It is assumed that the entity returned by applying the operator to a predicate satisfies certain uniqueness and existence requirements. Existence requirements are formulated using a wide-scope existential quantifier in the sentence the iota appears in. Uniqueness requirements are treated as presuppositions about the context of the entailment and therefore they are formulated as conditions conjoined to the term of *t*, even if the iota appeared in *h*.

The ζ operator is used for designating an entity that represents the class of an *et* predicate. It is implemented as a function that takes a predicate and returns a fresh new entity. This entity replaces the predicate in the rest of the analysis. The name of the entity is determined by the name of the predicate in order to assure that different predicates to which ζ is applied will be represented by different entities.[7]

Lexical inferences that the human annotators marked, for example, *last* → *former*, are also taken to be part of the entailment context. Due to that, similarly to uniqueness requirements, they are formulated as conditions conjoined to the term of *t*. The formulation is done according to the type of the words/phrases that the lexical relation pairs. The data in (42), (43), (44) and

[6] Note that $\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x) \wedge \mathbf{loves}(x)(\mathbf{mary})) = \exists x.\mathbf{boy}(x) \wedge \mathbf{loves}(x)(\mathbf{mary})$ based on the definition of $\exists_{(et)t}$ given in Section 4.3.1. The platform reduces function applications of existential and universal quantification functions only when the formulas of t and h are translated to predicate calculus in Section 5.2.7.

[7] In practice the platform uses a hashing algorithm: the name of the predicate is hashed in order to generate a unique name for the entity that represents it. Predicates with different names will be given different hash codes and consequently be represented by entities with different names.

```
function tree-to-formula(element)
    if branching-node(element)
        if (type-of(element.left)=$\tau\sigma$ $\wedge$ type-of(element.right)=$\tau$)
            return define-function-app(tree-to-formula(element.left),
                                       tree-to-formula(element.right))
        else if (type-of(element.left)=$\tau$ $\wedge$ type-of(element.right)=$\tau\sigma$)
            return define-function-app(tree-to-formula(element.right),
                                       tree-to-formula(element.left))
        else
            return undefined
    else
        return element
end

function beta-reduce(term)
    reduced-term = reduce-step(term)
    while not-equal(reduced-term,term)
        term = reduced-term
        reduced-term = reduce-step(term)
    end
    return term
end
function reduce-step(term)
    if function-application(term)
        func = get-application-func(term)
        arg = get-application-arg(term)
        if lambda-abstraction(func)
            return get-abstraction-body(func)[
                    get-abstraction-var(func):=rename(func,arg)]
        else
            reduced-func = reduce-step(func)
            if not-equal(reduced-func,func)
                return define-function-app(reduced-func,arg)
            else
                return define-function-app(func,reduce-step(arg))
    else if lambda-abstraction(term)
        return define-lambda-abstraction(get-abstraction-var(term),
                    reduce-step(get-abstraction-body(term)))
    else
        return term
end
```

Figure 5.3: Pseudo-code of the *tree-to-formula* and *beta-reduce* Functions

(45) illustrate the conditions formulated for relations between $(et)t$, et, eet and $(et)et$ predicates respectively.

(42) a. $(et)t$: *John* → *Lennon*

b. $\exists x_e.\mathbf{john}_{(et)t}(=(x)) \wedge \mathbf{lennon}_{(et)t}(=(x))$

The entities that define the $(et)t$ quantifiers **john** and **lennon** are the same.

(43) a. et: *jumped* → *moved*

b. $\forall x_e.\mathbf{jump}_{et}(x) \rightarrow \mathbf{move}_{et}(x)$

The entities in the extension of the et predicate **jump** are also in the extension of **move**.

(44) a. eet: *kissed* → *touched*

b. $\forall x_e.\forall y_e.(\mathbf{kiss}_{eet}(x))(y) \rightarrow (\mathbf{touch}_{eet}(x))(y)$

The entities that are in the eet relation **kiss** are also in the eet relation **touch**.

(45) a. $(et)et$: *last* → *former*

b. $\forall A_{et}.\forall x_e.(\mathbf{last}_{(et)et}(A))(x) \rightarrow (\mathbf{former}_{(et)et}(A))(x)$

The entities in the restriction of a set by the $(et)et$ modifier **last** are also in the restriction of that set by **former**.

5.2.6 Lowering to First Order Logic

The processing explained above calculates terms for t and h. In order to complete the implementation of the model described in Chapter 4 in a computational framework, we need to check whether t's term entails h's term. This check is automated by using a first order theorem-prover. In this section we explain a heuristic procedure for lowering the higher-order representations of the platform to first-order formulas in order to use a theorem prover in this logic.

The system's lexicon is defined in higher-order logic, and as a result the terms calculated for t and h include higher-order expressions such as the function application $\mathbf{italian}_{(et)et}$ $(\mathbf{opposition}_{et})$. Therefore, these terms are in second-order logic. For feasibility reasons, it is convenient not to use higher-order terms directly, but to translate them to first-order logic. This allows us to employ a theorem prover that uses effective methods for proving theorems in this logic. As a result the platform can be responsive to the annotator. For more on the use of first-order logic in computational semantics see Blackburn and Bos (2005).

The component described in this subsection implements a procedure for lowering the terms of t and h to first order logic. It should be noted that we opted for a simple translation procedure as it was easy to implement while sufficiently powerful for our purposes. We did not encounter any case in which this procedure limited the functionality of the annotation platform. However, more sophisticated lowering procedures can be employed in the future if needed.

Order Lowering Procedure

The only higher-order expressions in the interpreted lexicon are the restrictive modifiers: MR for adjectives and adverbs, P_R for prepositions and GR_1/2 for gerunds, and the non-restrictive modifier NR for adjectives.[8] These are all predicates of type $(et)et$. For example, let us reconsider Example (40) from Chapter 4, repeated in (46) below:

(46) *t*: The head of the Italian opposition, Romano Prodi, was the last president of the European Commission.

h: Romano Prodi is a former president of the European Commission.

The analysis of the text contains the modification $\textbf{italian}_{(et)et}$ $(\textbf{opposition}_{et})$, in which the predicate $\textbf{italian}_{(et)et}$ is a higher order function that takes the et predicate $\textbf{opposition}_{et}$ as an argument.

The platform handles such cases by replacing each of these modifications with a predicate of type et. The name of the new predicate is determined by concatenating the names of all constants included in predicates participating in the modification. For example, the modification in (47a) is treated as shown in (47b).

(47) a. $\textbf{italian}_{(et)et}(\textbf{opposition}_{et})$

b. $\textbf{italian\_opposition}_{et}$

It should be clarified that $\textbf{italian\_opposition}_{et}$ does not entail $\textbf{opposition}_{et}$. These are arbitrary functions of type et and there is no relation between their extensions. The system is able to derive that an entity in the extension of *Italian opposition* is also in the extension of *opposition* due to the conjunction in the semantics of the restrictive modifier MR: $\lambda A_{et}.\lambda x_e.A(x) \wedge WORD_{(et)et}(A)(x)$. The order lowering procedure described here deals only with the expression $WORD_{(et)et}(A)$. Let us review the full analysis:

[8] The higher-order existential and universal quantification functions are treated separately in Section 5.2.7.

Italian/MR *opposition*/N

$$\begin{array}{ll|l}
= & (\lambda A_{et}.\lambda x_e.A(x) \wedge \mathbf{italian}_{(et)et}(A)(x))(\mathbf{opposition}_{et}) & \text{def. of MR and N} \\
= & \lambda x_e.\mathbf{opposition}_{et}(x) \wedge \mathbf{italian}_{(et)et}(\mathbf{opposition}_{et})(x) & \text{func. app.} \\
= & \lambda x_e.\mathbf{opposition}_{et}(x) \wedge \mathbf{italian\_opposition}_{et}(x) & \text{order lowering} \\
\subseteq & \lambda x_e.\mathbf{opposition}_{et}(x) & \text{analysis} \\
= & \mathbf{opposition}_{et} & \text{lambda abs.} \\
= & \textit{opposition}/\text{N} & \text{analysis}
\end{array}$$

A similar explanation can be given to the rest of the examples in this subsection.

The same algorithm applies when the predicate being modified is a complex predicate, as exemplified in the lowering of (48a) to (48b).

(48) a. $\mathbf{famous}_{(et)et}(\lambda x_e.\mathbf{black}_{et}(x) \wedge \mathbf{singer}_{et}(x))$

b. $\mathbf{famous\_black\_singer}_{et}$

In cases where the modifier or the modified predicate is a function application that takes arguments of type e, as in (49a) and (50a) respectively, the generated predicate is also a function application with the same entity arguments applied. Lowered versions appear in (49b) and (50b) respectively.

(49) a. $(\mathbf{in}_{e(et)et}(\mathbf{Brazil}_e))(\mathbf{walks}_{et})$

b. $\mathbf{in\_walks}_{eet}(\mathbf{Brazil}_e)$

(50) a. $\mathbf{famous}_{(et)et}(\lambda x_e.\mathbf{black}_{et}(x) \wedge \mathbf{president}_{eet}(\mathbf{us}_e)(x))$

b. $\mathbf{famous\_black\_president}_{eet}(\mathbf{us}_e)$

The order lowering function, *lower–order*, is defined according to the pseudo-code in Figure 5.4. The pseudo-code assumes the following functions in addition to those mentioned in Sections 5.2.3 and 5.2.4:

- *concat-names*(names[]) concatenates a list of strings with an underscore. For example, *concat-names*("famous","black","president") = "famous_black_president".
- *define-term*(name,arguments[]) defines an *et* term in which a constant with a requested name is applied to a list of arguments. For example, *define-term*("famous_black_president","us") = $\mathbf{famous\_black\_president}_{eet}(\mathbf{us}_e)$.
- *constant*(term) returns *true* if the given term is a constant word and $false$ otherwise.

- *type*(term) returns the type of a term.
- *name*(term) returns the name of a constant term.

Lowering applications of higher order modifiers requires a different strategy when lexical relations are defined between such modifiers. For example, consider the lexical relation defined between *last* and *former* in Section 4.4.2, repeated in (51) below:

(51) $\forall A_{et}.\forall x_e.\mathbf{last}_{(et)et}(A)(x) \rightarrow \mathbf{former}_{(et)et}(A)(x)$

The annotation indicates the lexical relation between the modifiers by universally quantifying over all modifiable predicates. There is no particular *et* predicate to which the higher order modifiers are applied and therefore lowering $\mathbf{last}_{(et)et}(A)$ and $\mathbf{former}_{(et)et}(A)$ to $\mathbf{last\_A}_{et}$ and $\mathbf{former\_A}_{et}$ respectively would fixate invalid names for the predicates in the lexical relation.

The platform handles such cases by duplicating the lexical inference for each *et* predicate that appears in *t* and *h* For example, assuming that some pair contains only two *et* predicates, $\mathbf{occasion}_{et}$ and $\mathbf{president}_{eet}(\mathbf{us}_e)$. The lexical relation between *last* and *former* of type (*et*)*et* formulated in (51) is treated as the two formulas shown in (52).

(52) a. $\forall x_e.\mathbf{last}_{(et)et}(\mathbf{occasion}_{et})(x) \rightarrow \mathbf{former}_{(et)et}(\mathbf{occasion}_{et})(x)$

b. $\forall x_e.\mathbf{last}_{(et)et}(\mathbf{president}_{eet}(\mathbf{us}_e))(x)$

$\rightarrow \mathbf{former}_{(et)et}(\mathbf{president}_{eet}(\mathbf{us}_e))(x)$

Thus, for each *et* predicate, the platform defines an implication in which this predicate is modified. Then all modifications are lowered to first order logic as described in Figure 5.4 and conjoined. This turns (51) into the conjunction in (53).

(53) $[\forall x_e.\mathbf{last\_occasion}_{et}(x) \rightarrow \mathbf{former\_occasion}_{et}(x)] \wedge$

$[\forall x_e.\mathbf{last\_president}_{eet}(\mathbf{us}_e)(x) \rightarrow \mathbf{former\_president}_{eet}(\mathbf{us}_e)(x)]$

In this way all lexical relations are lowered to first order logic.

The limitations of this order lowering procedure are apparent in examples like (54).

(54) *t*: John is a tall man and every man is a person and every person is a man.

h: John is tall person.

```
function lower-order(modifier-exp,modified-exp)
        new_pred_name = concat-names(
                extract-predicates-names(modifier-exp) $\cup$
                extract-predicates-names(modified-exp))
        new_pred_entities =
                extract-predicates-entity-args(modifier-exp) $\cup$
                extract-predicates-entity-args(modified-exp)
        return define-term(new-pred-name,new_pred_entities)
end

function extract-predicates-names(exp)
        if constant(exp) $\wedge$ (type(exp)=et $\vee$ type(exp)=eet)
            return {name(exp)}
        else if lambda-abstraction(exp)
            return extract-predicates-names(get-abstraction-body(exp))
        else if function-application(exp)
            return extract-predicates-names(get-application-func(exp)) $\cup$
                   extract-predicates-names(get-application-arg(exp))
        else
            return $\emptyset$
end

function extract-predicates-entity-args(exp)
        if function-application(exp)
            func = get-application-func(exp)
            arg = get-application-arg(exp)
            if constant-pred-based(func) $\wedge$ type(arg)=e
                return {arg} $\cup$ extract-predicates-entity-args(func)
            else
                return $\emptyset$
        else
            return $\emptyset$
end

function constant-pred-based(exp)
        if constant(exp) $\wedge$ (type(exp)=et $\vee$ type(exp)=eet)
            return true
        else if function-application(exp)
            return constant-pred-based(get-application-func(exp))
        else
            return false
end
```

Figure 5.4: Pseudo-code of the *lower–order* Function

The terms calculated for t and h are in (55):

(55) t: $\mathbf{man}_{et}(\mathbf{john}_e) \wedge (\mathbf{tall}_{(et)et}(\mathbf{man}_{et}))(\mathbf{john}_e) \wedge (\forall x_e.\mathbf{man}_{et}(x) \rightarrow \mathbf{person}_{et}(x)) \wedge (\forall x_e.\mathbf{person}_{et}(x) \rightarrow \mathbf{man}_{et}(x))$

h: $\mathbf{person}_{et}(\mathbf{john}_e) \wedge (\mathbf{tall}_{(et)et}(\mathbf{person}_{et}))(\mathbf{john}_e)$

And their lowered forms are in (56):

(56) t: $\mathbf{man}_{et}(\mathbf{john}_e) \wedge \mathbf{tall\_man}_{et}(\mathbf{john}_e) \wedge (\forall x_e.\mathbf{man}_{et}(x) \rightarrow \mathbf{person}_{et}(x)) \wedge (\forall x_e.\mathbf{person}_{et}(x) \rightarrow \mathbf{man}_{et}(x))$

h: $\mathbf{person}_{et}(\mathbf{john}_e) \wedge \mathbf{tall\_person}_{et}(\mathbf{john}_e)$

It is easy to see that the expression $\mathbf{tall\_person}_{et}(\mathbf{john}_e)$ that appears in h's lowered term does not appear in t's. Therefore it is not possible for the first-order theorem prover to prove that t's lowered term entails h's lowered term, although h can be shown to be a logical consequence of t in the formal model. This means that the platform is an incomplete proof system with respect to the semantic theory.

However, this is an artificial example that does not represent common language use. In fact, in our extensive work with the platform, we did not encounter any example where the annotation platform could not find a proof for a pair that can be shown to be a logical entailment in the semantic model. This brings us to the more general question of the soundness and completeness of the platform.

Soundness and Incompleteness Considerations

The components described in Sections 5.2.1–5.2.5 treated t and h exactly as they are analyzed in the formal model of Chapter 4. This includes parsing them into phrase structures, binding their words to the lexicon, handling iota instances and expressing lexical inferences. However, in order to prove that the term calculated for h logically follows from t the platform lowers the terms to first order logic using the procedure described above. The platform acts as a proof system and therefore we have to examine its soundness and completeness with respect to the formal model.

- A proof system is said to be sound with respect to a given semantics if any deductive consequence found by the system is a logical consequence in the semantics. This property indicates that the deductive system does not prove theorems that are not true in the semantics. In our case, the proof system is required to obtain a proof that the term calculated for h is a deductive consequence of the term of t only if t logically entails h in the formal model described in Chapter 4.

- A proof system is said to be complete with respect to a given semantics if any logical consequence in the semantics is a deductive consequence

> found by the system. This property indicates that the deductive system can prove any theorem that is true in the semantics. Thus completeness requires that any h that is logically entailed by a t in the formal model of Chapter 4 would be deducible from that t by the proof system.

Kruit (2013) proved that a proof system incorporating the order-lowering procedure described above is sound but incomplete. For the purpose of creating a corpus of annotated entailments this turned out to be a reasonable compromise. On the one hand, the proving stage can be automated and guaranteed to deduce only an h that is logically entailed by a t according to the truth-conditionality criterion in the formal model (soundness). This means that any annotated pair for which the term calculated for h is found to follow from the term calculated for t can be considered well-annotated. On the other hand, the platform will not be able to prove some entailments that are true in the formal model (incompleteness), as illustrated above in (54). However, as mentioned above, the extensive work that the platform has been put into showed us that its incompleteness is a minor limitation. In fact we did not encounter any pair that constitutes a true entailment in the formal model and was shown to be undeducible in the framework. Therefore this limitation did not influence the collection of data for the corpus. Kruit's proof is restated with minor adjustments in Appendix A.

5.2.7 Search for a Proof

The platform uses an off-the-shelf theorem prover to search for a proof that indicates whether the lowered term of t entails the lowered term of h. The architecture is modular and allows us to use a wide range of automatic theorem provers for first-order logic. The prover currently employed is McCune's (2010) *Prover9* as it was found reliable, fast and easy to work with.

Prover9's input format is given in predicate calculus and therefore the formulas are rewritten in this format. This translation is done according to a fixed mapping between expression formats. For example, functions written in the Currying convention in lambda calculus, such as $(\mathbf{meet}_{eet}(\mathbf{john}_e))(\mathbf{mary}_e)$, are rewritten in a multiple-argument format in predicate calculus: $\mathbf{meet}(\mathbf{john}, \mathbf{mary})$.[9] At this stage the platform also reduces function applications of existential and universal quantification functions: $\exists_{(et)t}(\lambda x_e.f_{et}(x))$ and $\forall_{(et)t}(\lambda x_e.f_{et}(x))$. The reductions are done based on the definitions of $\exists_{(et)t}$ and $\forall_{(et)t}$ given in Section 4.3.1, and result in first-order formulas with the standard existential and universal quantifiers. For example, $\exists_{(et)t}(\lambda x_e.\mathbf{boy}(x))$ is reduced to $\exists x.\mathbf{boy}(x)$.

If a proof is found then it is recorded in the platform for a future use and the annotator continues to a new pair; if a proof is not found then the annotator reexamines the inferential process by which the pair was annotated and verifies

[9] The transformation to predicate calculus and the integration with Prover9 were implemented by Benno Kruit. See Kruit (2013). The implementation is included in Kokke (2013b).

that all steps are marked using annotations. After adding/fixing annotations, the annotator requests the platform to reprocess the data and to start over the search for a proof. In this sense, the platform implements an *Annotating by Proving* methodology – only those pairs for which the platform could deduce the hypothesis from the text are considered well–annotated.

5.3 Limitations of the Platform

Some limitations of the annotation platform are summarized below:

- The translation from second-order-logic to first order results in the platform being an incomplete proof system with respect to the formal model. However, as mentioned above, in our extensive use of the platform, we have not encountered cases where incompleteness impairs in actual use.
- Since first-order logic is undecidable, it is not guaranteed that a theorem prover will identify that a proof cannot be found when it does not exist. Thus, for non-entailing pairs, the theorem prover may not halt its search. However, in our work with the platform we did not encounter such a case. Prover9 turned out to be very fast both in proving theorems and in identifying that no proof exists. In the future a model checker such as McCune's (2010) standalone *Mace4* can be executed in parallel with the theorem prover to potentially speed up the response to the annotator when no proof can be found.[10]

Despite these limitations, it is important to note that the soundness of the proof system guarantees that it will not generate a proof when t does not entail h in the semantic model. Thus positive pairs where an annotation is missing or incorrect, and negative pairs, will not be provable.

5.4 Annotating Using the Platform

We have thus far explored the pipeline that SemAnTE's annotation platform employs in processing a *t*–*h* pair. Let us now illustrate the annotation of a concrete example using this annotation platform. Consider example (46) repeated in (57) below:

(57) *t*: The head of the Italian opposition, Romano Prodi, was the last president of the European Commission.

h: Romano Prodi is a former president of the European Commission.

[10] A model checker searches for a model in which t is true and h is false. This technique might provide a relatively quick resolution when a proof cannot be found since it could be easier to construct such a countermodel than to exhaust the theorem prover search space.

Parsing and Annotating Automatically

As explained above, the first task in processing a pair is obtaining phrase structures for *t* and *h*. Subsequently, the heuristic described in Section 5.2.2 annotates words based on part-of-speech tags. These two steps result in the annotated parse tree in (58). Although the syntax contains many mistakes, the annotation is decently close to the final annotation after corrections.

(58) *t*: [[The/THE head/N_1] [of/P_R [[the/THE [Italian/MR opposition/N_1]], [Romano Prodi]/NP]]], [was/IS [[the/THE [last/MR president/N_1]] [of/P_R [the/THE [European/MR Commission/NP]]]]].

h: [Romano Prodi]/NP [is/IS [a/A [former/MR president/N_1]] [of/P_R [the/THE [European/MR Commission/NP]]]].

Manually Correcting the Syntax and Annotations

Human annotators review the syntax and annotations and correct them in order to allow the proof system to process the data and to find a proof indicating that *t* entails *h*.

In the case in point, the parser segmented the first part of the apposition incorrectly: it generated a constituent structure corresponding to *the Italian opposition, Romano Prodi*. This implies that Romano Prodi is the Italian opposition, although the sentence conveys that he is the *head* of the opposition. In addition, the parser created a structure *[The head] [of the Italian opposition]*, in which the prepositional phrase modifies the full noun phrase rather than the internal noun as in *[The [head [of the Italian opposition]]]*. The same issue also appeared in the phrase structure *[the last president] [of the European Commission]* that had to be changed to *[the [last [president [of the European Commission]]]]*. All of these issues are corrected manually by an annotator. At the end of the syntactic corrections, *t* and *h* look as in (59).

(59) *t*: [[The [head [of [the [Italian opposition]]]]], [Romano Prodi]], [was [the [last [president [of [the [European Commission]]]]]]].

h: [Romano Prodi] [is [a [former [president [of [the [European Commission]]]]]]].

With respect to annotations, the only annotations that needed to be fixed are those of *last* and *former*, which need to be annotated as non-restricting modifiers, NR, rather than restrictive, and *president*, *head* and *of* that need to be annotated as relational nouns, N_2, and as a relational preposition, OF, respectively.

The corrections of the syntax and annotations yield annotated parse trees as in (60).

(60) *t*: [[The/THE [head/N_2 [of/OF [the/THE [Italian/MR opposition/N_1]]]]] APP/WHO_A [Romano Prodi]/NP], [was/IS_{eq}[the/THE [last/NR [president/N_2 [of/OF [the/THE [European Commission]/N_1]]]]]].

h: [Romano Prodi]/NP [is/IS [a/A [former/NR [president/N_2 [of/OF [the/THE [European Commission]/N_1]]]]]].

In addition to that, annotators also need to indicate lexical inferences that license the entailment. In this case the inference *last* → *former* is indicated.

Once the parse trees and the annotations are fixed, the annotators can request the system to initiate a type consistency check and then to try to prove the entailment.

Running a Type Check

Type checking is needed in order to verify that the parse trees of t and h are semantically composable. In case of error, the platform points the annotator to the branching node in which function application is not possible. This requires the annotator to correct the annotations and then to rerun the type check. In case of success, the processing moves on to the next stages. Since in this example, t and h in (60) are properly annotated, they pass the type check.

Obtaining Terms

The platform uses a beta-reducer to generate terms for t and h. Subsequently, several applications of higher order functions are lowered:

- $\mathbf{european}_{(et)et}(\mathbf{commission}_{et}) \Rightarrow \mathbf{european\_commission}_{et}$
- $\mathbf{italian}_{(et)et}(\mathbf{opposition}_{et}) \Rightarrow \mathbf{italian\_opposition}_{et}$
- $\mathbf{last}_{(et)et}(\lambda x_e.((\mathbf{president}_{eet}(x2_e))(x_e))) \Rightarrow$ $\mathbf{last\_president}_{eet}(x2_e)$
- $\mathbf{former}_{(et)et}(\lambda x_e.((\mathbf{president}_{eet}(x2_e))(x_e))) \Rightarrow$ $\mathbf{former\_president}_{eet}(x2_e)$

In addition to that, the lexical inference *last* → *former* is replaced by a set of formulas in first order logic.

The terms of t and h are now translated to predicate calculus. as illustrated in Figures 5.5 and 5.6 respectively. A snippet of the entailment context which formulates the lexical inferences and uniqueness conditions projected from definite descriptions is given separately in (5.7).

Proving the FOL Entailment

The platform can now execute Prover9 to search for a proof that the term of *h* follows from the term of *t* combined with the entailment context. Prover9 was able to find such a proof and based on that the pair is declared as well-annotated.

5.5 Summary

SemAnTE's platform establishes a computationally assessable connection between entailments in natural language and semantic representations. The connection is drawn using a semi-automatic process that begins with automatic parsing and heuristic annotation. Then a stage of manual correction is performed by human annotators in order verify that the linguistic phenomena conceived in their understanding of the entailment are marked. Once the manual work is done and the trees pass a type check, the platform generates terms representing the text and hypothesis, and integrates lexical relations and contextual assumptions into them. These higher order representations are translated to first order logic in order to be amenable for automatic theorem proving. The platform is a sound proof system with respect to the formal model described in Chapter 4 but also incomplete. Soundness guarantees that a positive result from the theorem prover can be interpreted as an indication that the annotated syntactic configurations account for the entailment in the formal model.

```
exists x0 ((opposition(x0) & italian_opposition(x0)) & exists x1
    ((head(x1) & of_head(x0, x1)) & exists x2 ((commission(x2) &
     european_commission(x2)) & exists x3 ((x3=x1 & x3=
    Romano_prodi) & exists x4 (last_president(x2, x4) & x4=x3)))
    )).
```

Figure 5.5: Text Term

```
exists x0 ((commission(x0) & european_commission(x0)) & exists x1
     (former_president(x0, x1) & x1=Romano_prodi)).
```

Figure 5.6: Hypothesis Term

```
Uniqueness conditions:
all x0 all x1 (((commission(x0) & european_commission(x0)) & (
    commission(x1) & european_commission(x1))) -> x0=x1).
all x0 all x1 (((opposition(x0) & italian_opposition(x0)) & (
    opposition(x1) & italian_opposition(x1))) -> x0=x1).
all x0 all x1 ((exists x2 ((opposition(x2) & italian_opposition(
    x2)) & (head(x0) & of_head(x2, x0))) & exists x3 ((
    opposition(x3) & italian_opposition(x3)) & (head(x1) &
    of_head(x3, x1)))) -> x0=x1).

Implications:
all z0 (last_commission(z0) -> former_commission(z0)).
all z0 (last_european_commission(z0) ->
    former_european_commission(z0)).
all z0 (last_head(z0) -> former_head(z0)).
all z0 (last_opposition(z0) -> former_opposition(z0)).
all z0 (last_italian_opposition(z0) -> former_italian_opposition(
    z0)).
all z0 (exists x0 ((commission(x0) & european_commission(x0)) &
    last_president(x0, z0)) -> exists x0 ((commission(x0) &
    european_commission(x0)) & former_president(x0, z0))).
```

Figure 5.7: Entailment Context

CHAPTER 6

Corpus Creation Using the Proof-Based Platform

6.1 Overview

This chapter describes the creation of a new theory-based corpus of annotated entailments – Semantic Annotation of Textual Entailment (SemAnTE) 2.0 – using the proof-based annotation platform introduced in the previous chapter. The corpus is theory-based as all of its entailing and non-entailing pairs can be given a formal explanation using the explicit linguistic theory encoded in the semantic model that the annotation platform implements. The corpus consists of 600 annotated pairs in an positive-negative ratio of 2:1. It is freely available on the web.[1] The data collection process is explained and illustrated. Possible future developments are described at the end.

6.2 Data Collection

The implementation of a semantic model of entailment in a logically sound annotation platform allowed us to created a theory-based annotated corpus of entailment data. The corpus demonstrates the fragment of English to which the model can be applied in order to explain inferences. As part of this work we could test and fine tune the model and the platform that implements it.

A useful feature of the proof-based approach is that it provides an immediate indication whether an inferential path that an annotator marked is a valid inference semantically. This allows our human annotators to easily spot cases

[1] See: `http://logiccommonsense.wp.hum.uu.nl/resources`

in which an incomplete or invalid inferential path was annotated. In addition to that, the platform allows each annotator to mark linguistic phenomena according to the inferential path in her/his subjective interpretation of the data. The platform ascertains that these annotations support a formal inferential path (or lack thereof) from the analysis of the text to the hypothesis.

These features of the proof-based annotation platform allowed us to define an annotation methodology termed *Annotating-By-Proving*. This procedure dictates that an entailing pair is considered well-annotated only if the marked annotations allow the platform to generate a proof that the analysis of the text based on the implemented theory entails the analysis of the hypothesis. An extension of Annotating-By-Proving covers non-entailing pairs as well: A non-entailing pair is considered well-annotated only if the platform is unable to generate a proof for it based on the annotations while on the other hand, it is able to generate a proof for a minimally different entailing pair that is annotated as similarly as possible. We elaborate on these methodologies below.

Importantly, Annotating-By-Proving, which was enabled by implementing the semantic model in a sound annotation platform, is our method for ascertaining that the corpus is indeed theory-based.

In the first part of the corpus we used the RTE as a source of positive examples. The datasets of the RTE provide a readily available resource of entailment data that aim to demonstrate natural language inferences. However, it should be noted that our main goal is to create a theory-based corpus. Thus we accept the limitations of the semantic theory that is employed and due to that the pairs that are included in our corpus are restricted in their syntactic and semantic complexity. Therefore, we simplify the syntax and the inferential phenomena of RTE examples such that our semantic model would be able to explain the entailment relation. Only in a negligible number of five cases, we used RTE pairs as-is.

In the second part we created couplets of positive and negative pairs which show minimal contrasts between an intuitively positive and negative judgment on the entailment that the formal model can account for. These couplets illustrate the predictive power of the model, as they demonstrate that the model distinguishes between positive and negative examples in line with the human judgments.

6.2.1 Creating Positive Examples

The first step in using the annotation platform to create positive examples was to examine the extent to which our semantic model can cover the inferential phenomena included in RTE entailments. This was done by human annotators who browsed the data and documented their understanding of inferences.[2] We found that although inferences stemming from modification phenomena and

[2] Some of the annotators were already thoroughly familiar with the RTE based on their work on SemAnTE 1.0.

simple quantification were frequently used in the inferential paths created by our human annotators, it was rarely the case that they were the sole phenomena explaining the inference. In the vast majority of the entailments at least one step in the inferential path that the human annotators created stemmed from a phenomenon that is not modeled in the lexicon. Furthermore, in many cases the syntax of the RTE data was more flexible or complex than what the model supports. For example, consider (61):

(61) *t*: Meanwhile, the inquiries spearheaded by Milan magistrates Antonio di Pietro and Gherardo Colombo began to focus on Italy's private and public business sectors.

h: Antonio di Pietro is a magistrate.

The title construction *Milan magistrates Antonio di Pietro and Gherardo Colombo* allows us to infer that Antonio di Pietro and Gherardo Colombo *are* Milan magistrates. Then comes an inference from plural to singular based on which we derive that Antonio di Pietro is a Milan magistrate. Finally, by restrictivity of the modification construction *Milan magistrate*, we can infer that he is a magistrate. Thus the hypothesis follows. What we see here is an inferential path that incorporates inferences stemming from the modeled phenomena of appositive and restrictive modifications intermediated by a plural to singular inference that the lexicon does not account for.

In addition, notice the temporal adverb *Meanwhile* that modifies the main verb *began*. In the current paradigm of annotating (static) parse trees it is not possible to mark non-adjacent constituents as modifiers. Thus the fronted adjunct cannot be marked as a modifier of the verb. It is also not possible to analyze the adverb as a sentence modifier since there is no modifier of an element of type t in the lexicon.[3]

In cases like the above, where the inferential path created by the annotator is only partially covered by the phenomena modeled in the lexicon, or where the syntactic configuration cannot be captured by the present model of syntax, we let the annotators create new, simpler pairs, based on the original ones. The simplified pairs present non-trivial entailments and aim to preserve the natural quality of the data.

In practice, the example in (61) was simplified to (62):

(62) *t*: The inquiries spearheaded by Milan magistrate Antonio di Pietro focused on Italy's private business sectors.

h: Antonio di Pietro is a magistrate.

The fronted adjunct was removed and the plural to singular inference was avoided. This simplified version was annotated as in (63).

[3] All modifiers in the lexicon are defined as modifiers of predicates of type *et*.

(63) *t*: [[The/THE [inquiries/N_1 [APP/WHO_R [spearheaded/V_1 [by/P_R [[DET/A [Milan/MR magistrate/N_1]] [APP/WHO_A [Antonio di Pietro]/NP]]]]]]]] [[focused on]/V_2 [[Italy/NP 's/POSS] [private/MI [business/MR sectors/N_1]]]]

h: [[Antonio di Pietro]/NP [is/IS [a/A magistrate/N_1]]]

The annotation platform generated a proof that accounts formally for the inference from *t* to *h*. Thus, following the Annotating-By-Proving approach, it is considered well-annotated. In Section 6.4 we give more examples of simplifications that were applied to pairs from the RTE.

Another technique that we employed was to rewrite RTE pairs in order to generate entailment pairs that are focused on modification and quantification. This technique gives the human annotators more freedom in creating the examples.

Let us illustrate this with the example in (64), which was simplified to (65).

(64) *t*: The crew of Apollo 11, Neil Armstrong, Buzz Aldrin and Michael Collins, and other early astronauts were named "Ambassadors of Exploration" in a ceremony at the Smithsonian National Air and Space Museum.

h: Aldrin was one of the astronauts honored by NASA at the Smithsonian National Air and Space Museum.

(65) *t*: The astronaut, Neil Armstrong, was named "Ambassador of Exploration" in the ceremony held by NASA at the Smithsonian National Air and Space Museum.

h: Armstrong was given a title of honor in a ceremony held by NASA.

The annotator also indicated three lexical relations:

- *Neil Armstrong* → *Armstrong*
- *named* → *given*
- *Ambassador of Exploration* → *a title of honor*

In (65) we have an inferential path that stems from several modeled phenomena. Informally, the appositive modification in *The astronaut, Neil Armstrong, was...* allows us to infer that *Neil Armstrong was....* Based on the first lexical relation defined above we can infer that *Armstrong was....* Furthermore, the other lexical relations license the inference from *named "Ambassador of Exploration"* to *given a title of honor*. The entailment also includes an inference from a definite description – *the ceremony held by NSA* to an indefinite *a ceremony* *held by NSA*. Finally, restrictivity licenses the inference from *held by NASA at the Smithsonian National Air and Space Museum* to *held by NASA*.

This example was annotated as shown in (66). The proof system found a formal deductive process from *t* to *h* based on the annotations marked.

(66) *t*: [[[The/THE astronaut/N_1] [APP/WHO_A [Neil/NP Armstrong/NP]]] [was/IS [[[named/V_2 [Ambassador/NP [of/NP Exploration/NP]]] [in/P_I [[the/THE ceremony/N_1] [APP/WHO_A [held/V_1 [by/P_R NASA/NP]]]]]] [at/P_R [the/THE [Smithsonian/NP [National/NP [Air/NP [and/NP [Space/NP Museum/NP]]]]]]]]]

h: [Armstrong/NP [was/IS [[given/V_2 [a/A [title/N_1 [of/P_R [DET/EMPTYDET honor/N_1]]]]] [in/P_I [[a/A ceremony/N_1] [APP/WHO_A [held/V_1 [by/P_R NASA/NP]]]]]]]

6.2.2 Creating Couplets of Positive and Negative Examples

In addition to the creation of positive examples as described above, we created couplets of positive and negative pairs. These cases intuitively show a "minimal contrast" between valid entailments and maximally similar pairs of sentences that do not constitute valid entailments. We concentrate on such contrasts that are accounted by our semantic model: entailments that are proven by the platform *vis a vis* similar but invalid entailments that the system does not prove. These pairs show the predictive power of the model and can be used in the future for learning possible ways of distinguishing between positive and negative pairs automatically. The pairs had to be created by the annotators, as it is rarely the case that a negative example from the RTE can be explained by the semantics of modification solely.[4]

Our methodology for creating couplets that show a minimal contrast between a positive and a negative example was as follows. Firstly, a pair which demonstrates an intuitively positive example is created. The pair is composed in a way that its inferential path relies on modification and simple quantification phenomena. A human annotator applies Annotating-By-Proving to the pair and verifies that the platform is able to deduce the term calculated for the hypothesis from the term of the text. Secondly, a pair which demonstrates an intuitively negative example (i.e. no entailment) is created. This pair shares the same text with the positive one and has a minimally adjusted hypothesis. The minimal adjustments made to the hypothesis are meant to block any inferential process from the text. The positive and negative pairs are annotated as close as possible and an annotator verifies that the platform cannot prove the negative one. At the end of this process the pairs are considered well-annotated.

[4] Such a rare example appears in Pair 908 of RTE 2 development set: *Warner is the world's largest media and internet company* $\not\Rightarrow$ *Time Warner is the world's largest company.*

Following this methodology, we assume that we have control for the reasons negative examples cannot be proven by the platform. In each couplet, it is a certain semantic phenomenon that licenses a required local inference in the positive example and this inference is not possible in the negative one due to its adjusted hypothesis.

Furthermore, the minimal contrast between the proving positive example and the not-proving negative one allows us to conclude that the lack of proofs for negative examples is not due to the incompleteness of the proof system. The fact that the system finds a proof for the positive examples and not for the minimally adjusted negative ones substantiates this conclusion. In addition, we can assume that it is not some external reason such as unmodeled world knowledge that does not allow the system to prove a negative pair. For if it was the reason then the positive pair would not have been proven as well.

Note that for each negative pair it should be possible to construct a model in which the text is true and the hypothesis is false. This can be done automatically using a model checker or manually by a human annotator. Constructing a model automatically without creating a minimal contrast between a proving pair and a minimally adjusted negative one that is annotated as close as possible, does not guarantee that the lack of proof for the negative pair is due to the reason we aim to isolate. Constructing a model manually is labor intensive and does not help our annotators to verify their annotations.

A number of examples of couplets of positive and negative pairs are given below. These examples show three main strategies for creating such couplets.

- A restrictive modifier displacement

(67) *t* Vlado Taneski, the prolific Macedonian killer convicted for the murder of several elderly women, was a journalist.

$h - pos$ Vlado Taneski was a Macedonian journalist.

$h - neg$ Vlado Taneski was a prolific journalist.

In this couplet, the word *prolific* is annotated as a restrictive modifier and the word *Macedonian* as an intersective one. Therefore, if Vlado Taneski is a *prolific Macedonian killer*, he is *Macedonian*, and since in addition he is a *journalist*, it is possible to infer that he is a *Macedonian journalist.* However, it is not possible to infer that he is *prolific* (in general). Since this is a restrictive modifier, he is only a *prolific Macedonian killer.* Thus we cannot derive that he is a *prolific journalist* as requested by the hypothesis of the negative pair. In both hypotheses the tree structures and all annotations are the same, except for the usage of restrictive vs. intersective modifier. The standard semantics of restrictive and intersective modification that was presented in Chapter 4 immediately accounts for

the negative example because the truth-conditionality criterion does not hold here: we can construct a model where t is true, and $h - neg$ is false. For example, consider a model M whereby:
$[\![\textit{Vlado Taneski}]\!]^M = \mathbf{vt}$
$[\![\textit{Bob Woodward}]\!]^M = \mathbf{bw}$
$[\![\textit{journalist}]\!]^M = \{\mathbf{vt}, \mathbf{bw}\}$
$[\![\textit{prolific journalist}]\!]^M = \{\mathbf{bw}\}$
$[\![\textit{Macedonian}]\!]^M = \{\mathbf{vt}\}$
$[\![\textit{prolific Macedonian journalist}]\!]^M = \{\}$

- A lexical relation under the scope of restrictive modification

(68) t Jose Mujica, who lives on a ramshackle farm and gives away most of his pay, is considered the poorest president in the world.

$h - pos$ Jose Mujica is considered the poorest president in the world.

$h - neg$ Jose Mujica is considered the poorest politician in the world.

A lexical relation of *president* → *politician* is indicated for the negative pair in order to provide it with the lexical information that an inferential path from the text to the hypothesis requires. Yet, even if every president is a politician, being the poorest president does not imply being the poorest politician. The standard semantics of restrictive modification accounts for such negative example. For instance, in the model M given below t is true and $h - neg$ is false.
$[\![\textit{Jose Mujica}]\!]^M = \mathbf{jm}$
$[\![\textit{Manik Sarkar}]\!]^M = \mathbf{ms}$
$[\![\textit{president}]\!]^M = \{\mathbf{jm}\}$
$[\![\textit{politician}]\!]^M = \{\mathbf{jm}, \mathbf{ms}\}$
$[\![\textit{poorest president in the world}]\!]^M = \{\mathbf{jm}\}$
$[\![\textit{poorest politician in the world}]\!]^M = \{\mathbf{ms}\}$
$[\![\textit{considered the poorest president in the world}]\!]^M = \{\mathbf{jm}\}$
$[\![\textit{considered the poorest politician in the world}]\!]^M = \{\mathbf{ms}\}$

It should be noted here that any restrictive modifier could be used instead of a superlative. Superlatives are usually easy to work with since they are intuitive restrictives but only their restrictiveness property is important for our purposes.

- Switching subject and object

(69) *t* Scandal creator Shonda Rhimes will host the 44th President at her Los Angeles home.

h − pos Shonda Rhimes will host a politician.

h − neg A politician will host Shonda Rhimes.

In this case the hypotheses differ only in the way the subject and object are switched. Hence, the fact that the proof system cannot prove the hypothesis of the negative pair from the text can be safely attributed to the incorrect order of composition of elements in its tree structure – the verb *host* should be combined with *a politician* first and then with *Shonda Rhimes*. The motivation for creating pairs using this strategy is that it generates couplets of positive and negative pairs that have the same, or at least very close, lexical overlap. Thus the pairs are indistinguishable based on lexical measures and a deeper analysis is required in order to classify them correctly.

6.3 Figures on the Annotated Data

SemAnTE 2.0 contains 600 pairs: 400 positive and 200 negative. The structure of the corpus is as follows.

- Positive Pairs: contains 200 positive pairs created by simplifying entailments from RTE 1–4. Each pair in the corpus has its original RTE version as well as the version annotated and proven in the annotation platform.
- Couplets of Positive-Negative Pairs: contains 200 couplets of positive and negative pairs. In each couplet the text is shared between the pairs and the hypotheses differ. Both pairs are fully annotated and it is verified that only the positive one is proven in the platform.

6.4 Common Simplifications of RTE Examples

Let us examine some representative examples of simplified RTE pairs. For each pair we describe the versions before and after simplifications and explain the reasons for making them.

- (70) Original:

  t: The international humanitarian aid organization, Doctors Without Borders/Medecins Sans Frontieres (MSF), continues to treat victims of violence in all locations where it is present in Darfur.

  h: Doctors Without Borders is an international aid organization.

  Simplified:

  t: The international humanitarian aid organization, Doctors Without Borders, Medecins Sans Frontieres (MSF), treats victims of violence in all locations that exhibit violence in Darfur.

  h: Doctors Without Borders is an international aid organization.

  The necessary simplifications in this example are syntactic in nature. The first issue is in the infinitival clause *Doctors Without Borders . . . continues to treat. . . .* the subject of the internal clause *to treat. . .* does not appear in the syntax. In linguistic theories this is analyzed as a null pronoun named *PRO*, whose antecedent is determined by the subject of a control predicate, *continues* in the case in point. The annotators simplified *continues to treat* to *treats*. An additional point arises regarding the anaphora in *locations where it is present*. Since anaphora is not modeled, the phrase was simplified to avoid it.

- (71) Original:

  t: The San Diego Padres ace, Jake Peavy, was hurt in an 8-5 loss to the St. Louis Cardinals.

  h: Jake Peavy is a player of the San Diego Padres.

  Simplified:

  t: The ace of the San Diego Padres, Jake Peavy, was hurt in a loss to the Saint Louis Cardinals.

  h: Jake Peavy is a player of the San Diego Padres.

  In this example we have a noun phrase, *San Diego Padres*, taking the position of a modifier of a relational noun *ace*. This syntax is not supported as there is no way to express the relation between these expressions using the lexicon. The annotator simplified this to *The ace of the San Diego Padres.*

- (72) Original:

  t: The Zulu are an African ethnic group of about 11 million people who live mainly in KwaZulu-Natal Province, South Africa.

  h: The Zulus live in Kwazulu-Natal Province.

  Simplified:

  t: The Zulu are an African ethnic group of about 11 million people who mainly live in KwaZulu-Natal Province, which is in South Africa.

  h: The Zulus mainly live in Kwazulu-Natal Province.

  This example contains a modifier of a preposition – *live mainly in KwaZulu-Natal Province* – which the lexicon does not offer a way to treat. The natural interpretation of the sentence is that among all places where the Zulu live, KwaZulu-Natal Province is the main one. The annotator moved the modifier to an adverbial position and made it *mainly live in KwaZulu-Natal Province*, which conveys the same meaning. The annotator interpreted *mainly* as a non-restrictive modifier, since it implies that not all Zulus live in KwaZulu-Natal. Thus, logically, it is not possible to infer *mainly live in X* → *live in X* as requested by the hypothesis. Due to that the annotator also qualified *live* in the hypothesis with *mainly.* Another issue here concerns the comma construction *KwaZulu-Natal Province, South Africa.* This is a sort of locative apposition, as it conveys that KwaZulu-Natal Province is *located in* South Africa, rather then being equal to it. The annotator simplified the construction accordingly to express this meaning.

To conclude, many entailments from the RTE to which the model is applied require some degree of simplification. This is either because their syntax contains unsupported constructions, for example fronted adjuncts and infinitival clauses, or because the phenomena currently modeled in the lexicon are too limited and cannot explain an inferential path from the text to the hypothesis. For example, entailments that involve a plural to singular inference cannot be accounted for. These simplifications can be reduced by modeling more semantic phenomena. and supporting more syntactic constructions.

6.5 Annotation Work Load

The composition and annotation of new pairs in SemAnTE 2.0 were mainly done in parallel to the development of the annotation platform. In the final stages of the development, it is estimated that each pair of sentences (either positive or negative) took about 15 minutes to create and annotate. However, it is expected that in order to obtain a more substantial entailment corpus based on the same methodology, more development work would be needed on the semantic theory and the platform implementing it. The amount of work that would be needed depends on the complications of the phenomena involved.

6.6 Discussion

The corpus of SemAnTE 2.0 described in this chapter demonstrates the extent to which the semantic theory of Chapter 4 implemented in the annotation platform of Chapter 5 can be applied to entailment data. The pairs in the corpus are composed and annotated following systematic procedures: (1) a methodology of Annotating-By-Proving which assures that positive pairs are logic entailments in the semantic theory and (2) an extension of this methodology for creating couplets of positive and negative pairs generates pairs with a minimal contrast that is sufficient for having contrasting judgments on the entailment relation that the semantic theory explains. In this way we created a theory-based corpus: a corpus in which all entailment relations or lack thereof are explained by an explicit linguistic theory.

There are three main differences between this work and the annotation works done by Garoufi (2007), Bentivogli et al. (2010a) and Sammons et al. (2010), described in Chapter 2: (1) in SemAnTE the inferences or lack thereof are explained by a formal semantic theory, (2) the annotation work is proof-based, and (3) SemAnTE is restricted in the syntactic and semantic phenomena that it manifests. These issues are connected: formal semantic representations that can be described in first-order logic can be linked to proof theory, thus allowing a computational assessment of the inferential path being modeled. But on the other hand, these theories are restricted both in terms of the syntactic constructions they support and in the range of inferences that they explain.

It should be noted that despite these limitations, SemAnTE is not a test-suite for a semantic theory. Unlike the data in the FraCaS test suite (Cooper et al., 1996), it is based as much as possible on examples that appear in an unrestricted corpus (RTE). Pairs from the RTE are simplified to the minimal extent that allows the semantic theory to explain them. New pairs are inspired by RTE pairs and also aim to represent inferences in natural language.

A practical advantage of the proposed annotation approach is that it provides an immediate indication whether the inferential path that an annotator marked is a valid inference semantically. This allows our human annotators to easily spot cases in which an incomplete or invalid inferential path was annotated. Furthermore, each annotator can annotate the subjective inferential path by which (s)he understands the entailment. The requirements from annotators are to understand the phenomena they mark and to follow the methodology of annotating-by-proving and its extension. The computational framework takes care of substantiating the existence of an inferential paths that they mark (or lack thereof).

The work is in line with the proposal of Sammons et al. (2010) to address natural language entailment with annotation work in a restricted domain of linguistic phenomena, to evaluate it and later on to extend the domain. By modeling more semantic phenomena and supporting more syntactic constructions the scope of inferences that the model accounts for can be extended, thus covering a larger fragment of English while still maintaining the proposed methodologies and ascertaining high precision in the annotation work.

6.7 On Further Directions for Entailment Recognition

The results reported here are a natural starting point for developing a theory-based entailment recognizer. This can be done by decoupling the two main components of the annotation platform: the user interface and the proof system. Then the proof system can be used for building a theory-based entailment recognizer.

However, considerable adaptations are required for making this a viable direction. The culprit is that the proof system requires correct parse trees augmented with semantic annotations and exact specifications of lexical relations. If any of these data is missing or incorrect, the proof system cannot provide a reliable indication whether the semantic model explains the inference. As part of an annotation platform, this is a desired outcome. However, in order to achieve a theory-based entailment recognizer, further research is needed in order to develop robust, recall-oriented, automatic analyzers of the syntactic and semantic annotations that are required by the proof system.

This line of inquiry would need to target a better balance between precision and recall, sacrificing accuracy for robustness. One strategy to achieve it might be to incorporate degrees of freedom in the proof system's operation. Instead of working with a single parse tree per sentence, which has to be accurately structured, a recognizer could allow the parser to generate several alternative parse trees for each sentence. It is expected that many of these trees will be erroneous, but these would fail to pass a type-check and therefore are easily ignored. This approach increases the chances of obtaining at least one parse tree that the semantic model supports, thus increasing robustness. The risk is that negative examples may become provable if the parser assigns them linguistically implausible structures. This could decrease precision - but if the degrees of freedom are relatively small, we can assume that such cases should be uncommon.

Degrees of freedom can also be added within the automatic annotation algorithm. Similarly to assigning multiple trees to each sentence, the automatic annotation procedure can be extended to derive several possible annotations. This would increase the chances of the output being sufficiently close to the annotations produced by human annotators. Therefore, positive examples are more likely to be processed successfully without human intervention. However, as in the previous case, such an approach would reduce the precision of the system.

The degrees of freedom in any of the system's components would have to be parametrized. Controlling these parameters should allow achieving a balance between precision and recall. As in many similar circumstances in natural language processing, optimal balance could be sought after by machine learning techniques.

CHAPTER 7

Conclusions

This thesis has proposed a theory-based paradigm for investigating textual entailment. This paradigm is premised on the assumption that entailment recognizers could be made more accurate if an explicit linguistic theory explains at least some of the data that they are designed to cover. We have focused on the initial step in materializing this paradigm: developing a computational framework that enables an annotation methodology for obtaining theory-based entailment data.

The first question that was investigated is: to which linguistic phenomena is it feasible and expedient to apply a theory-based paradigm? Our initial hypothesis was that inferences stemming from restrictive modification and its intersective and appositive varieties are suitable for an exploration of the proposed approach. These semantic phenomena were hypothesized to be common in natural language entailment.

To validate this hypothesis, we used the corpora of the Recognizing Textual Entailment (RTE) task. This task had established the contemporary paradigm in natural-language processing for evaluating entailment recognizers. It consists of several thousand entailments designed to represent types of inference that are common in natural unrestricted language. Accordingly, RTE data were deemed appropriate for testing the proposed theory-based paradigm.

An annotation project that probes the frequency of modification-based inferences was carried out by two human annotators on the entailing pairs from the RTE 1–4 datasets. The annotators marked all instances of modification phenomena that played a role in their interpretation of the entailment relation between the text and hypothesis. A quantitative analysis of the annotations indicated that inferences triggered by modification are found in about 80% of the positive entailments in RTE 1–4. The annotated datasets were subsequently released as a corpus, entitled Semantic Annotation of Textual Entailment (SemAnTE) 1.0, which is publicly available.

The process of developing SemAnTE 1.0 revealed challenges in annotating inference-triggering phenomena and verifying the accuracy of the annotations marked. Since the analysis is informal and involves translating an intuitive entailment judgment to a sequence of concrete linguistic phenomena, an annotator may miss one or more such structure or annotate it inaccurately. Inter-annotator agreement (IAA) checks did not prove to be a useful tool for detecting such annotation errors. In many cases, divergent annotations occurred because annotators follow different legitimate inferential paths in recognizing an entailment relation. IAA checks show low agreement in such cases, where both annotations are valid.

Based on the conclusions from the work on SemAnTE 1.0 and our goal to investigate textual entailment in a theory-based paradigm we developed a new methodology of entailment annotation, referred to as Annotating-By-Proving (ABP). This methodology relies on an inference model for analyzing textual entailment, which is based on a standard formal semantic model. It is hypothesized that by following the ABP methodology, an annotation platform can ascertain that the annotated data are explained by the semantic theory and provide human annotators with a feedback on whether their annotations account for the entailment. In order to check the usability of this thesis, we implemented a computational framework relying on the ABP methodology.

The proposed framework implements a semantic model that boosts a typed lexicon that encodes the treatment of modification phenomena as well as simple universal and existential quantification. Words are assigned semantic meanings by being bound to lexical items, and complex expressions are assigned meanings recursively based on the meanings of their parts (compositionality). Function application in lambda calculus simplifies the sentential terms obtained for the text and hypothesis. The model defines *Semantic Entailment* by applying the truth-conditionality criterion to the sentential terms of the text and hypothesis of a given pair. The linguistic phenomena are analyzed using set-theoretical concepts. The model assumes simple representations, which are applied to entailment data following standard tree tagging conventions. Arguably, this property of compositional semantics is important for the model's learnability. It also allows to begin with a theory that models a small set of linguistic phenomena and then augment it to cover increasingly complex syntactic constructions and semantic inferences.

An annotation platform that implements the model was developed in order create a computational framework for generating theory-based entailment data. The platform automatically processes an entailment pair using the following components: (1) a syntactic parser and a part of speech tagger for obtaining parse trees annotated with syntactic category labels, (2) a heuristic for using part of speech tags for binding lexical tokens to the lexicon of the system, (3) a type consistency checker for validating that function application is possible for the trees of the text and the hypothesis, (4) a beta reducer for simplifying the sentential terms of the text and the hypothesis, (5) an order lowering algorithm for reducing the terms in higher-order logic to first-order, and (6) a theorem prover designed to generate, whenever possible, a formal proof that the term calculated for the text entails the term of the hypothesis. Human annotators mark lexical relations and correct parsing and annotation errors resulting from imperfections in the automatic tools used for components (1) and (2) above. The tools and techniques employed by the platform link together the syntactic-semantic tagging of parse trees in natural text, model theory, and proof theory.

A crucial consideration in developing the platform was to make it sound with respect to the semantic theory and reasonably responsive, even at the expense of completeness. With respect to every pair of sentences, the system's soundness renders the proof it generates a reliable indication that a theoretical proof can be derived within the semantic model adopted. This allowed us to materialize ABP: an entailing pair can be judged well-annotated only if the annotations can serve to generate a proof between the representations of the text and the hypothesis. An extension of this methodology covers non-entailing (negative) pairs as well: a non-entailing pair is considered well-annotated only if the platform is unable to generate a proof based on the annotations; while on the other hand, it is able to generate a proof for a minimally different entailing pair that is annotated as similarly as possible. The minimal contrast between an entailing pair and its proof, on the one hand, and a non-entailing pair and the lack of proof, on the other, warrants the conclusion that the lack of proof for the non-entailing example is not due to an annotation error or the incompleteness of the system.

The platform served to evaluate the applicability of the semantic model to entailment data. To this end, a new corpus of annotated entailments was created, named SemAnTE 2.0. The corpus comprises two parts: (1) entailing (positive) pairs which were created by simplifying RTE examples, avoiding unsupported syntactic constructions and semantic phenomena, and (2) couplets of new minimally contrasting positive and negative pairs sufficient for eliciting contrasting entailment judgments within the semantic model. Due to this methodology, all entailment judgments in SemAnTE 2.0 are explained by the semantic model. The dataset consists of 600 pairs in a positive-negative ratio of 2:1 and is freely available on the web.

It has been demonstrated how a computational framework that enables Annotating-By-Proving is employed by human annotators to create entailment data accounted by an explicit linguistic theory. The conclusion is that computational implementations of standard formal semantic theory make it feasible to develop a theory-based paradigm of textual entailment. Relatively simple representations encoded in semantic models have sufficient explanatory power to account for a wide variety of inferences in natural language. The availability of an explicit linguistic theory that accounts for the data created in this paradigm holds promise for developing better performing entailment recognizers.

APPENDIX A

Soundness and Non-completeness

Chapter 5 described a proof system that incorporates a lower-ordering procedure. This procedure takes a formula in higher order logic and lowers it to first order logic. This is done by iterating over instances of function applications $A_{(et)et}(B_{et})$ and replacing them with et predicates. The soundness and non-completeness of this procedure was proved by Kruit (2013). This appendix restates the proof with some adjustments.[1]

Soundness and Completeness

Let us recall the definitions of soundness and completeness of a proof system with respect to a given semantics:

- Soundness: a deductive (proof) system is said to be sound with respect to a given semantics if any deductive consequence found by the system is a logical consequence in the semantics.
- Completeness: a deductive (proof) system is said to be complete with respect to a given semantics if any logical consequence in the semantics is a deductive consequence found by the system.

Soundness indicates that the proof system does not prove theorems that are not valid in the semantics. Completeness indicates that the proof system can prove any theorem that is valid in the semantics.

[1] Kruit's proof is slightly adjusted here for the sake of presentation. It is reproduced with Kruit's kind permission.

Goal

Let us examine the treatment of a t-h pair using the platform of Chapter 5 in comparison to a theoretical analysis of such a pair using the semantic model of Chapter 4.

In both cases, t and h are parsed into phrase structures and annotated with semantic phenomena by binding their words and phrases to the lexicon. Sentential meanings for t and h are obtained using a compositional analysis. We can assume that both the theoretical analysis using the semantic model and the computational analysis using the annotation platform yield the same terms for t and h.

The analysis continues by generating a formal proof that the term calculated for h can be deduced from the term calculated for t. Due to practical considerations of decidability, the platform lowers the higher-order terms in t and h to first order prior to the execution of an off-the-shelf theorem prover in first-order logic.

Our goal is to show that as a result of the order lowering procedure, the annotation platform is a sound proof system with respect to the semantic model but incomplete. Consider the proof system as a proving process comprised of two steps, given the terms of t and h:

(a) Lowering the order of higher-order predicates in t and h to first order. At the end of this procedure we obtain t' and h' in first order logic.

(b) Searching for a proof that h' can be deduced from t'.

Let us name this proof system $P1$ and define a proof system, $P2$, whose proving process includes only step (b). By definition, if t and h are both in first order logic, $P2$ returns the same result as $P1$. Otherwise $P2$ returns 0. We assume that $P2$ is sound and prove that $P1$ is also sound but incomplete.

Proof

The proof is structured in several iterations over a logical system with an increasingly complex language. We begin with a simple system comprised of constants only and end up with a system that contains the lexicon presented in Chapter 5.

Let C be the set of (logical and non-logical) constants:

$$\begin{aligned} C = \{&x_\sigma : x \text{ is a non-logical constant} \wedge \sigma \in \{e, et, eet, eeet, \ldots\}\} \cup \\ &\{1, 0, \forall_{(et)t}, \exists_{(et)t)}, \wedge_{ttt}, \vee_{ttt}, \rightarrow_{ttt}, \leftrightarrow_{ttt}, \neg_{tt}\} \end{aligned}$$

Let FOL be the set of first order expressions:

$$\begin{aligned} FOL = C \cup & \\ & \{x_e : x \text{ is a variable}\} \cup \\ & \{a_{\sigma\tau}(b_\sigma) : a, b \in FOL\} \cup \\ & \{a_{(et)t}(\lambda x_e.b_t) : a, b \in FOL\} \end{aligned}$$

Languages with Constants Only

Let us define L_1 as the smallest superset of FOL that includes function applications of $(et)et$ constants. Formally:

$$\begin{aligned} L_1 = C \cup & \\ & \{x_e : x \text{ is a variable}\} \cup \\ & \{m_{(et)et} : m \text{ is a constant}\} \cup \\ & \{a_{(et)t}(\lambda x_e.b_t) : a, b \in L_1\} \cup \\ & \{a_{\sigma\tau}(b_\sigma) : a, b \in L_1 \wedge \sigma\tau = (et)et \rightarrow b \text{ consists of only constants}\} \end{aligned}$$

Algorithm

Let φ be a formula in L_1. Let n be the number of instances of a function application $a_{(et)et}(b_{et})$ in φ.

- Let $\varphi_0 = \varphi$
- For each instance of a function application $a_{(et)et}(b_{et})$ in φ_i:
  - Create a fresh constant c_{et}. The name of c_{et} is a concatenation of the names of the constants in $a_{(et)et}(b_{et})$.
  - Let $\varphi_{i+1} = \varphi_i$ where $a_{(et)et}(b_{et})$ is replaced by c_{et}.
- Return φ_n

For example, the function application $(tall_{(et)et}(student_{et}))(john_e)$ is replaced by $tall\_student_{et}(john_e)$.
All instances of $a_{(et)et}$ in φ are eliminated when the n-th iteration of the algorithm is performed. Thus φ_n is in FOL.

Soundness

Let $\varphi_i = t_i \rightarrow h_i$ be a formula that is lowered to φ_{i+1}. We assume that φ_{i+1} is a tautology and prove that φ_i has to be a tautology as well.

If φ_i is not a tautology, then there is a model M that does not satisfy it. We can now construct a model M' without loss of generality such that $[\![c_{et}]\!]^{M'}$

$= [\![a_{(et)et}(b_{et})]\!]^M = [\![a_{(et)et}]\!]^M([\![b_{et}]\!]^M)$. But then model M' does not satisfy φ_{i+1}, in contrast to the assumption that φ_{i+1} is a tautology. Thus φ_i is also a tautology.

This proves that in the lowering step from φ_i to φ_{i+1}, if the latter is a tautology then the former has to be a tautology as well. By applying the same reasoning n times iteratively we can prove that if the output of the algorithm, φ_n, is a tautology then the input of the algorithm, φ_0, is a tautology either.

Now let us use this result to prove that $P1$ is sound given that $P2$ is sound. Assume a proof system $P1$ that given a *t*-*h* pair, lowers the formula $\varphi = t \rightarrow h$ to a first order formula $\varphi' = t' \rightarrow h'$ using the algorithm described above and then executes a sound proof system $P2$ to prove φ'. If a proof is found, then based on $P2$'s soundness, φ' is a tautology. Consequently, based on the result from above, it follows that φ is a tautology as well and therefore $P1$ is sound.

Non-completeness

Assume that x_e and $a_{(et)et}$ are arbitrary constants and let p_{et} and q_{et} be constant functions that satisfy: $\forall M.\ [\![p_{et}]\!]^M = \{x_e\} = [\![q_{et}]\!]^M$. Assume further that $t = (a_{(et)et}(p_{et}))(x_e)$, $h = (a_{(et)et}(q_{et}))(x_e)$ and $\varphi_0 = t \rightarrow h$. We will show that φ_0 is a tautology and φ_1 is not.

Based on our assumptions, $\forall M.\ [\![t]\!]^M = [\![(a_{(et)et}(p_{et}))(x_e)]\!]^M = ([\![a_{(et)et}]\!]^M ([\![p_{et}]\!]^M))(x_e) = ([\![a_{(et)et}]\!]^M([\![q_{et}]\!]^M))(x_e) = ([\![a_{(et)et}(q_{et}))(x_e)]\!]^M = [\![h_i]\!]^M$. It follows that φ_i is a tautology. However, lowering it to φ_1 results in replacing $a_{(et)et}(p_{et})$ and $a_{(et)et}(q_{et})$ by the constants c_{et} and d_{et} respectively. Therefore, $\varphi_1 = c_{et}(x_e) \rightarrow d_{et}(x_e)$. We can construct a model M such that $c_{et}(x_e) = 1$ and $d_{et}(x_e) = 0$. Thus M does not satisfy φ_1 and as a result φ_1 is not a tautology. This shows a case in which φ_0, the input of the algorithm, is a tautology while φ_1, the output, is not.

We can now use this result to show that $P1$ is incomplete. φ_1 is not a tautology and therefore $P2$, which is assumed to be sound, cannot prove it. It follows that $P1$ cannot prove φ_0 although it is a tautology. Hence $P1$ is incomplete.

Languages with Abstractions

Let us now extend L_1 into L_2 by introducing abstractions in the modifier argument position. L_2 is the smallest superset of FOL that includes $(et)et$ constants applied to abstractions or et constants. Formally:

$L_2 = C \cup$

$\{x_e : x \text{ is a variable}\} \cup$

$\{m_{(et)et} : m \text{ is a constant}\} \cup$

$\{a_{(et)t}(\lambda x_e.b_t) : a, b \in L_2\} \cup$

$\{a_{\sigma\tau}(b_\sigma) : a, b \in L_2 \wedge \sigma\tau = (et)et \rightarrow b$ consists of only constants$\} \cup$

$\{m_{(et)et}(\lambda x_e.b_t) : m, b \in L_2 \wedge m$ is a constant $\wedge x$ is the only free variable in $b\}$

Algorithm

Let φ be a formula in L_2. Let n be the number of instances of a function application $m_{(et)et}(\lambda x_e.b_t)$ in φ.

- Let $\varphi_0 = \varphi$.
- For each instance of a function application $m_{(et)et}(\lambda x_e.b_t)$ in φ_i:
  - Create a fresh constant c_{et}. The name of c_{et} is a concatenation of the names of the constants in b_t.
  - Let $\varphi_{i+1} = \varphi_i$ where $m_{(et)et}(\lambda x_e.b_t)$ is replaced by $m_{(et)et}(c_{et})$.
- Return φ_n

All instances of $m_{(et)et}(\lambda x_e.b_t)$ in φ are eliminated when the n-th iteration of the algorithm is performed. Thus φ_n is in L_1.

Soundness

Let $\varphi_i = t_i \rightarrow h_i$ be a formula that is lowered to φ_{i+1}. We assume that φ_{i+1} is a tautology and prove that φ_i has to be a tautology as well.

If φ_i is not a tautology, then there is a model M that does not satisfy it. We can now construct a model M' without loss of generality such that $\forall x_e$. $[\![c_{et}(x_e))]\!]^{M'} =[\![b_t]\!]^M$. But then M' does not satisfy φ_{i+1}, in contrast to the assumption that φ_{i+1} is a tautology. Thus φ_i is also a tautology.

By applying the same reasoning n times iteratively we can prove that if the output of the algorithm, φ_n, is a tautology then the input of the algorithm, φ_0, is a tautology either. The rest of the proof is as shown for languages with only constants.

Non-completeness

Non-completeness is proven in the same way as in the case of languages with only constants.

Languages with Free Variables and Complex Predicates

Let us now extend L_2 into L_3 by introducing free variables and more complex predicates. Formally, L_3 is the smallest superset that contains:

$$\begin{aligned} L_3 = C \cup & \\ & \{x_e : x \text{ is a variable}\} \cup \\ & \{m_{(et)et}, m_{e(et)et} : m \text{ is a constant}\} \cup \\ & \{a_{(et)t}(\lambda x_e . b_t) : a, b \in L_3\} \cup \\ & \{a_{\sigma\tau}(b_\sigma) : a, b \in L_3\} \cup \\ & \{m_{(et)et}(\lambda x_e . b_t) : m, b \in L_3 \ \} \end{aligned}$$

Algorithm

Let φ be a formula in L_3. Let n be the number of instances of a function application $a_{(et)et}(b_{et})$ in φ.

- Let $\varphi_0 = \varphi$.
- For each instance of function application $a_{(et)et}(b_{et})$ in φ_i:
  - Let $v_0 \ldots v_m$ be the m free variables of type e in $a_{(et)et}(b_{et})$
  - Create a fresh constant of type $e \ldots et$ with m+1 e-s: $c_{e \ldots et}$. The name of c_{et} is a concatenation of the names of the constants in $a_{(et)et}(b_{et})$.
  - Let $\varphi_{i+1} = \varphi_i$ where $a_{(et)et}(b_{et})$ is replaced by $c_{e \ldots et}$ applied to $v_0 \ldots v_m = (\ldots(c_{e \ldots et}(v_0)) \ldots (v_m))$
- Return φ_n

For example, when x_e is a free variable,

- $(quickly_{(et)et}(eat_{et}(x_e)))(john_e)$ is replaced by $(quickly\_eat_{et}(x_e))(john_e)$
- $((in_{e(et)et}(x_e))(jump_{et}))(john_e)$ is replaced by $(in\_jump_{et}(x_e))(john_e)$

All instances of $a_{(et)et}(b_{et})$ in φ are eliminated when the n-th iteration of the algorithm is performed. Thus φ_n is in FOL.

Soundness

Let $\varphi_i = t_i \to h_i$ be a formula that is lowered to φ_{i+1}. We assume that φ_{i+1} is a tautology and prove that φ_i has to be a tautology as well.

If φ_i is not a tautology, then there is a model M that does not satisfy it. We can now construct a model M' without loss of generality such that $[\![(\ldots(c_{e \ldots et}(v_0)) \ldots (v_m))]\!]^{M'} = [\![a_{(et)et}(b_{et})]\!]^M = [\![a_{(et)et}]\!]^M([\![b_{et}]\!]^M)$. But then model M' does not satisfy φ_{i+1}, in contrast to the assumption that φ_{i+1} is a tautology. Thus φ_i is also a tautology.

By applying the same reasoning n times iteratively we can prove that if the output of the algorithm, φ_n, is a tautology then the input of the algorithm, φ_0, is a tautology either. The rest of the proof is as shown for languages with only constants.

Non-completeness

Non-completeness is proven in the same way as in the case of languages with only constants.

L_3 is the largest superset of FOL we explored, and it is sufficiently expressive to define the lexicon introduced in Section 4.3 in it.

Summary

In Section 5.2 we presented a proof system which for practical reasons included an algorithm that lowers higher-order predicates to first order. In this appendix we proved that the proof system is sound with respect to the semantics defined in Chapter 4 and incomplete. The proof contains several iterations over an increasingly complex logical system – starting with a simple system comprised of a language with constants only and ending up with a system that contains the whole lexicon presented in Section 4.3.

APPENDIX B

Resources

This appendix describes the format in which the corpora of SemAnTE 1.0 and SemAnTE 2.0 are released. The corpora are freely available for research.[1]

B.1 SemAnTE 1.0

SemAnTE 1.0 was created using GATE Developer (Cunningham et al., 2011). As described in Section 3.3.2, the corpus contains the development and test sets of RTE 1–4 annotated with appositive, restrictive and intersective modification in cases where the phenomenon licensed the recognition of entailment by the human annotator. Only positive pairs were annotated. The corpus is released as a ZIP file containing the annotated datasets in GATE's XML serialisation format.[2] In addition to that, the compressed file contains documents describing the annotation guidelines, annotation schemes, annotation decisions, and a conference paper about the resource (Toledo et al., 2012).

A set of GATE annotation schemes were defined according to the syntactic expressions of appositive, restrictive and intersective modification. These schemes were used by the human annotators to mark the various instances of these phenomena. Table B.1 lists the annotation schemes assigned to each semantic phenomenon.

When a phenomenon is annotated in a *t*-*h* pair in SemAnTE 1.0, the operation involves two steps: (1) - marking the span of text in *t* which is the syntactic expression of the modification phenomenon by using a correlative annotation

[1] See: `http://logiccommonsense.wp.hum.uu.nl/resources`

[2] See: `https://gate.ac.uk/userguide/sec:corpora:output`

Modification Phenomenon	Syntactic Expressions	Annotation Schemes
Appositive	Appositions, Titles, Non-Restrictive Relative Clauses	apposition, title, rel_clause
Restrictive	Restrictive Modification	r_modification
Intersective	Conjunctions, Restrictive Relative Clauses	conjunction, rel_clause

Table B.1: GATE Annotation Schemes

scheme (for example, a title annotation scheme for a title construction), (2) - marking the span of text in h which corresponds to the output of the inferential step that the modification phenomenon licenses. The annotation in h is done using a dedicated *reference_to* scheme for indicating references. This marks the phenomenon and links between its syntactic expression in t and in h.

Figures B.1 and B.2 illustrate the XML files that define the annotation schemes for marking conjunctions and references respectively. The schemes specify fields that an annotator has to fill in when marking a span of text. For conjunctions, for annotator is required to separate the elements being conjoined (between two to five elements). In addition to that, in order to link between the syntactic expression of a phenomenon in t and in h, the annotator uses the same serial id when filling in the field *construction_id* in the phenomenon scheme marked in t and in the reference scheme marked in h. All schemes have an optional field for including comments in free text.

Example

Recall Example (23) from Chapter 3, repeated in (73) below with its analysis.

(73) *t*: The anti-terrorist court found two men guilty of murdering Shapour Bakhtiar and his secretary Sorush Katibeh, who were found with their throats cut in August 1991.

h: Shapour Bakhtiar died in 1991.

This entailment can be explained by the following inferential process:

- Initially, an inference from the relative clause *Shapour Bakhtiar and his secretary Sorush Katibeh, who were found with their throats cut in August 1991* to *Shapour Bakhtiar and his secretary Sorush Katibeh were found with their throats cut in August 1991* (appositive modification).

- Then, an inference from *August 1991* to *1991* (restrictive modification of *1991* by *August*).

```
<?xml version="1.0"?> <schema xmlns="http://www.w3.org/2000/10/
    XMLSchema">
   <!-- XSchema definition for Syntactic Constructions -->
   <element name="conjunction">
             <complexType>
                    <attribute name="E1" type="xs:string" use="
                        required"/>
                    <attribute name="E2" type="xs:string" use="
                        required"/>
                    <attribute name="E3" type="xs:string" use="
                        optional"/>
                    <attribute name="E4" type="xs:string" use="
                        optional"/>
                    <attribute name="E5" type="xs:string" use="
                        optional"/>
                    <attribute name="construction_id" type="xs:
                        integer" use="required"/>
                    <attribute name="comment" type="xs:string"
                        use="optional"/>
             </complexType>
       </element>
</schema>
```

Figure B.1: Annotation Scheme for Conjunctions

```
<?xml version="1.0"?> <schema xmlns="http://www.w3.org/2000/10/
    XMLSchema">
   <!-- XSchema definition for Syntactic Constructions -->
   <element name="reference_to">
             <complexType>
                    <attribute name="construction_id" type="xs:
                        integer" use="required"/>
                    <attribute name="comment" type="xs:string"
                        use="optional"/>
             </complexType>
       </element>
</schema>
```

Figure B.2: Annotation Scheme for References

- Finally, an inference from *Shapour Bakhtiar and his secretary Sorush Katibeh* to *Shapour Bakhtiar* (intersective modification).

By combining these three local inferences, it is possible to conclude *Shapour Bakhtiar was found with his throat cut in 1991.* Additional world knowledge is required to infer that *found with his throat cut* entails *died*; given that, the entailment can be fully validated.

Figures B.3 and B.3 show the annotation tags added to *t* and *h*, respectively, and serve both as boundary markers for this phenomenon and as pointers to the annotation content tags. Figures B.5 and B.6 present the annotation content tags that detail the conjunction in *t* and the reference to it from *h*.

```
The anti-terrorist court found two men guilty of murdering
<Node id="11404" />Shapour Bakhtiar and his secretary Sorush
Katibeh <Node id="11453" />, who were found with their throats
cut in August 1991
```

Figure B.3: Annotation Tags in (73)-*t*

```
<Node id="11511" />Shapour Bakhtiar<Node id="11527" />
died in 1991
```

Figure B.4: Annotation Tags in (73)-*h*

B.2 SemAnTE 2.0

SemAnTE 2.0 was created using the annotation platform described in Chapter 5. The corpus has two parts as explained in Chapter 6: positive pairs and couples of positive and negative pairs. The corpus is released as a ZIP file containing two sub-directories – Pos and Pos-Neg – that correspond to the two parts of the corpus. Each pair is described using several text files: (1) - a general description of the data, (2) - annotations of the text and hypothesis, and (3) - in case of a positive pair, a formal proof that shows how the text formally entails the hypothesis. These files are explained below.

- <pair_id>_description.txt – contains the text and hypothesis of the given pair before and after simplification. This is a text file in YAML format with two sections: *Original* and *Simplified*, each of which specifies a text string and an hypothesis string. In the case of POS-RTE, the *Original* section also includes a reference to the RTE corpus from where this pair is taken.

- <pair_id>_annotation.txt – contains the annotations of the simplified text and hypothesis. The file is divided into several parts:

```
<Annotation Id="1833" Type="conjunction" StartNode="11404"
    EndNode="11453">
       <Feature>
              <Name className="java.lang.String">E2</Name>
              <Value className="java.lang.String">his secretary
                  Sorush Katibeh</Value>
       </Feature>
       <Feature>
              <Name className="java.lang.String">construction_id
                  </Name>
              <Value className="java.lang.String">2</Value>
       </Feature>
       <Feature>
              <Name className="java.lang.String">E1</Name>
              <Value className="java.lang.String">Shapour
                  Bakhtiar</Value>
       </Feature>
</Annotation>
```

Figure B.5: Conjunction Annotation Details

```
<Annotation Id="1834" Type="reference_to" StartNode="11511"
    EndNode="11527">
       <Feature>
              <Name className="java.lang.String">construction_id
                  </Name>
              <Value className="java.lang.String">2</Value>
       </Feature>
</Annotation>
```

Figure B.6: Reference-To Annotation Details

- Entailment - indicates whether it is a positive or a negative pair: *true* indicates a positive pair and *false* a negative one.
- Text and Hypothesis: include the bracket structure, annotated trees and word-level information of the text and hypothesis respectively. The trees are printed using a customized bracket notation as follows:
  - Nodes: (<node label>:<id> <left sub-tree> <right sub-tree>)
  - Leaves: (<leaf annotation>:<id> [leaf text])

  For example, the tree: (NP:12 (A:3 [an]) (NP:11 (N:4 [oil]) (N:5 [giant]))) is headed by the node with id 12, labeled as *NP*, whose left sub-tree is the leaf with id 3, *an*, annotated as *A*, and its right sub-tree is the node with id 11, labeled as *NP*. The latter is the parent of two leaves annotated as $N - oil$ and *giant* whose ids are 4 and 5 respectively. Note that the ids are unique identifiers of tree elements (nodes and leaves).

  – Subsumptions - a set of ordered pairs that specify the lexical relations that were marked between nodes in the tree of the text and corresponding ones in the tree of the hypothesis.

- <pair_id>_proof.txt – contains the formal logical proof found by Prover9 between the semantic term of the text and that of the hypothesis. This file is included only for positive entailment pairs. It is generated by capturing prover9's standard output.

Pairs from Pos-Neg are included with either the *_pos* or *_neg* string in their file name to indicate whether these are the positive or negative variant of the couple respectively.

The corpus ZIP file also contains the definition of the lexicon whose items were used for annotating the trees. The lexicon file specifies the semantic term associated with each lexical item.

Example

Consider Pair 2 from Pos-RTE. The description file, *0002_description.txt* contains the reference to Pair 166 from the development set of RTE 1, the original text and hypothesis, and the simplified ones. The content of the file is shown in Figure B.7. The annotation file, *0002_annotation.txt*, contains the annotated trees of the text and hypothesis. The annotated text tree is shown in Figure B.8. Lastly, since this pair is a positive entailment pair, the corpus includes a the proof found by Prover9 based on the annotations. Figure B.9 presents the input provided to Prover9 and Figure B.10 shows the proof that Prover9 found.

```
Source: RTE 1 test set; Pair: 166.

Original:
    Text: The controversy-racked oil giant Shell has named a new
         head of finance in an effort to calm nervous shareholders.
    Hypothesis: Shell is an oil giant.

Simplified:
    Text: The controversy-racked oil giant Shell has named a new
         head of finance in an effort which calmed nervous
         shareholders.
    Hypothesis: Shell is an oil giant.
```

Figure B.7: Description File

```
[[[The [controversy-racked [oil giant]]] [APP Shell]] [has [[
    named [a [new [head [of [DET finance]]]]]] [in [an [effort [
    which [calmed [DET [nervous shareholders]]]]]]]]]]
(Merger:54 (S:45
    (NP:44
      (THE:1 [The])
      (NP:43
        (MR:2 [controversy-racked])
        (NP:42
          (N_1:3 [oil])
          (N_1:4 [giant]))))
    (Extract:53
      (WHO_A:52 [APP])
      (NP_D:5 [Shell])))
  (VP:35
    (V_AUX:6 [has])
    (VP:40
      (VP:39
        (V_2:7 [named])
        (NP:38
          (A:8 [a])
          (Merger:48
            (MR:9 [new])
            (NP:37
              (N_1:10 [head])
              (PP:24
                (P_R:11 [of])
                (Extract:47
                  (A:46 [DET])
                  (N_1:12 [finance])))))))
      (PP:33
        (P_R:13 [in])
        (Merger:51
          (A:14 [an])
          (NP:26
            (N_1:15 [effort])
            (SBAR:31
              (WHO_R:16 [which])
              (VP:29
                (V_2:17 [calmed])
                (Extract:50
                  (A:49 [DET])
                  (NP:28
                    (MR:18 [nervous])
                    (N_1:19 [shareholders])))))))))))
```

Figure B.8: Annotated Text

```
============= INPUT =============

formulas(assumptions).
(all x0 all x1 (oil_giant(x0) & controversyracked_oil_giant(x0) &
     oil_giant(x1) & controversyracked_oil_giant(x1) -> x0 = x1)
    ).
(all x0 all x1 ((exists x2 (oil_giant(x2) &
    controversyracked_oil_giant(x2) & x0 = x2 & x0 = Shell)) & (
    exists x3 (oil_giant(x3) & controversyracked_oil_giant(x3) &
     x1 = x3 & x1 = Shell)) -> x0 = x1)).
(exists x0 (oil_giant(x0) & controversyracked_oil_giant(x0) & (
    exists x1 (x1 = x0 & x1 = Shell & (exists x2 (head(x2) & (
    exists x3 (finance(x3) & of_head(x3,x2))) & h_1330210326(x2)
     & named(x2,x1))) & (exists x6 (effort(x6) & (exists x7 (
    shareholders(x7) & nervous_shareholders(x7) & calmed(x7,x6))
    ) & h_996092596(x6,x1))))))).
end_of_list.

formulas(goals).
(exists x0 (oil_giant(x0) & x0 = Shell)).
end_of_list.

============= end of input =============
```

Figure B.9: Input to Prover9

```
============= PROOF =============

3 (exists x0 (oil_giant(x0) & controversyracked_oil_giant(x0) & (
    exists x1 (x1 = x0 & x1 = Shell & (exists x2 (head(x2) & (
    exists x3 (finance(x3) & of_head(x3,x2))) & h_1330210326(x2)
     & named(x2,x1))) & (exists x6 (effort(x6) & (exists x7 (
    shareholders(x7) & nervous_shareholders(x7) & calmed(x7,x6))
    ) & h_996092596(x6,x1))))))) # label(non_clause). [
    assumption].
4 (exists x0 (oil_giant(x0) & x0 = Shell)) # label(non_clause) #
    label(goal). [goal].
7 oil_giant(c1). [clausify(3)].
9 c2 = c1. [clausify(3)].
10 Shell = c2. [clausify(3)].
11 c1 = Shell. [copy(10),rewrite([9(2)]),flip(a)].
12 -oil_giant(x) | Shell != x. [deny(4)].
15 oil_giant(Shell). [back_rewrite(7),rewrite([11(1)])].
16 $F. [ur(12,b,11,a(flip)),rewrite([11(1)]),unit_del(a,15)].

============= end of proof =============
```

Figure B.10: Prover9's Proof

APPENDIX C

Analyzing Processes of Inference using SemAnTE 1.0

This appendix is based on Toledo et al. (2012).

Abstract

This appendix describes a use case of SemAnTE 1.0 in which it serves for analysing the inferential processes generated by the BIUTEE system (Stern and Dagan, 2011). The corpus is used as a gold standard of modification phenomena that license inferences in the RTE, and the processes generated by BIUTEE are evaluated against them. The results of this evaluation are reported and indicate that BIUTEE rarely uses inferences stemming for modification phenomena to recognize an entailment.

Goal

Our goal is to examine the volume of instances in which a modification phenomenon is annotated in SemAnTE and BIUTEE relied on an inference that stems from this phenomenon to predict the entailment relation.

BIUTEE employs a rule-base of syntactic rules for a wide range of manually defined transformations (Lotan, 2012). Some of these rules were created specifically for deriving inferences that stem from appositive and intersective modification. Thus, we can examine in which pairs that include annotations of intersective or appositive modification in SemAnTE where treated by BIUTEE with corresponding rules that treat these phenomena.[1]

Results

We ran BIUTEE in two comparable configurations as described below and extracted information on rule applications from its log files. The analysis was done on the positive pairs in the corpora of RTE 1–3.

- C1 - Basic configuration with only syntactic rules included.
- C2 - Resource-based configuration with lexical and lexical/syntactic rules based on WordNet, FrameNet, VerbOcean, Catvar and DIRT, as well as the syntactic rules.

Table C.1 presents the coverage of instances of annotated intersective and appositive modification by rule applications in BIUTEE. P_{Ann} stands for pairs containing an annotation of appositive or intersective modification in SemAnTE, and $P_{Rule-Cx}$ stands for pairs that BIUTEE, running in configuration *Cx*, processed by applying a rule of appositive or intersective modification on the span of text of an annotated phenomenon in SemAnTE. $P_{\#}$ indicates the number of entailment pairs in each set (P_{Ann} or $P_{Rule-Cx}$ respectively) and $P_{\%}$ indicates the portion of a set in the total amount of positive entailment pairs in that corpus (in percents). On average, BIUTEE processes only 4.52% of the entailment cases based on rules that correspond to the phenomena annotated in SemAnTE.[2]

Table C.1: Coverage of annotated phenomena by BIUTEE's rule base.

Pairs set	RTE 1		RTE 2		RTE 3	
	$P_{\#}$	$P_{\%}$	$P_{\#}$	$P_{\%}$	$P_{\#}$	$P_{\%}$
P_{Ann}	210	53	241	60	239	58
$P_{Rule-C1}$	2	1	37	9	3	1
$P_{Rule-C2}$	4	1	47	12	15	4

In both configurations, the results indicate that BIUTEE's rule base is rarely used for processing the entailment patterns annotated in our corpus: 1% on RTE 1, 9-12% on RTE 2 and 1-4% on RTE 3.

[1] BIUTEE handles all modifiers as restrictive and as a result there is no specific rule that indicates an inference that stems from restrictivity per se.

[2] The marginal effect of the knowledge resources used in C2 on the overall accuracy of BIUTEE is in line with the results of ablation studies that were performed on various RTE systems (see Bentivogli et al. (2009) for more information).

The conclusion is that BIUTEE handles the annotated instances of modification phenomena using other transformational devices that it employs. The proofs generated by the system have very little in common with the inferential processes that the annotators of SemAnTE created in their own understanding of the entailments.

APPENDIX D

References to RTE Examples

Abstract

This appendix provides the references to all RTE examples that were used in this thesis.

RTE Examples

RTE 1 Dev Set, Pair: 19

t: Researchers at the Harvard School of Public Health say that people who drink coffee may be doing a lot more than keeping themselves awake – this kind of consumption apparently also can help reduce the risk of diseases.

h: Coffee drinking has health benefits.

RTE 1 Dev Set, Pair: 85

t: The country's largest private employer, Wal-Mart Stores Inc., is being sued by a number of its female employees who claim they were kept out of jobs in management because they are women.

h: Wal-Mart sued for sexual discrimination.

RTE 1 Dev Set, Pair: 140

t: A senior coalition official in Iraq said the body, which was found by U.S. military police west of Baghdad, appeared to have been thrown from a vehicle.

h: A body has been found by U. S. military police.

RTE 1 Dev Set, Pair: 166

t: The controversy-racked oil giant Shell has named a new head of finance in an effort to calm nervous shareholders.

h: Shell is an oil giant.

RTE 1 Dev Set, Pair: 561

t: The incident in Mogadishu, the Somali capital, came as U.S. forces began the final phase of their promised March 31 pullout.

h: The capital of Somalia is Mogadishu.

RTE 1 Dev Set, Pair: 567

t: Prime Minister Silvio Berlusconi was elected March 28 with a mandate to reform Italy's business regulations and pull the economy out of recession.

h: The Prime Minister is Silvio Berlusconi.

RTE 1 Dev Set, Pair: 579

t: The anti-terrorist court found two men guilty of murdering Shapour Bakhtiar and his secretary Sorush Katibeh, who were found with their throats cut in August 1991.

h: Shapour Bakhtiar died in 1991.

RTE 1 Dev Set, Pair: 581

t: To the world, M. Larry Lawrence, the new U.S. emissary to Switzerland who hosted President Clinton on his Southern California vacation, will be known as Mr. Ambassador.

h: Larry Lawrence is the head of the U.S. Embassy in Switzerland.

RTE 1 Dev Set, Pair: 908

t: Warner is the world's largest media and internet company.

h: Time Warner is the world's largest company.

RTE 1 Test Set, Pair: 994

t: Mr. Conway, Iamgold's chief executive officer, said the vote would be close.

h: Mr. Conway said the vote would be close.

RTE 1 Test Set, Pair: 1383

t: The crew of Apollo 11, Neil Armstrong, Buzz Aldrin and Michael Collins, and other early astronauts were named "Ambassadors of Exploration" in a ceremony at the Smithsonian National Air and Space Museum.

h: Aldrin was one of the astronauts honored by NASA at the Smithsonian National Air and Space Museum.

RTE 2 Dev Set, Pair: 45

t: The international humanitarian aid organization, Doctors Without Borders/Medecins Sans Frontieres (MSF), continues to treat victims of violence in all locations where it is present in Darfur.

h: Doctors Without Borders is an international aid organization.

RTE 2 Dev Set, Pair: 135

t: The watchdog International Atomic Energy Agency meets in Vienna on September 19.

h: The International Atomic Energy Agency holds a meeting in Vienna.

RTE 2 Dev Set, Pair: 147

t: Meanwhile, the inquiries spearheaded by Milan magistrates Antonio di Pietro and Gherardo Colombo began to focus on Italy's private and public business sectors.

h: Antonio di Pietro is a magistrate.

RTE 2 Dev Set, Pair: 154

t: Clonaid said, Sunday, that the cloned baby, allegedly born to an American woman, and her family were going to return to the United States Monday, but where they live and further details were not released.

h: Clonaid announced that mother and daughter would be returning to the US on Monday.

RTE 2 Dev Set, Pair: 220

t: Two persons were injured in dynamite attacks perpetrated this evening against two bank branches in this northwestern, Colombian city.

h: Two bank branches were attacked with dynamite.

RTE 2 Dev Set, Pair: 611

t: The book contains short stories by the famous Bulgarian writer Nikolai Haitov.

h: Nikolai Haitov is a writer.

RTE 2 Test Set, Pair: 241

t: The San Diego Padres ace, Jake Peavy, was hurt in an 8-5 loss to the St. Louis Cardinals.

h: Jake Peavy is a player of the San Diego Padres.

RTE 2 Test Set, Pair: 273

t: U.S. officials have been warning for weeks of possible terror attacks against U.S. interests.

h: The United States has warned a number of times of possible terrorist attacks.

RTE 2 Test Set, Pair: 365

t: Nixon was impeached and became the first president ever to resign on August 9th 1974.

h: Nixon was the first president ever to resign.

RTE 2 Test Set, Pair: 410

t: The head of the Italian opposition, Romano Prodi, was the last president of the European Commission.

h: Romano Prodi is a former president of the European Commission.

RTE 2 Test Set, Pair: 424

t: Microsoft Corp., on Thursday, posted higher quarterly earnings as revenue rose 12 percent, but its shares fell after the world's largest software market said current quarter sales would fall below Wall Street expectations.

h: Microsoft showed revenue growth.

RTE 2 Test Set, Pair: 750

t: The Zulu are an African ethnic group of about 11 million people who live mainly in KwaZulu-Natal Province, South Africa.

h: The Zulus live in Kwazulu-Natal Province.

RTE 3 Dev Set, Pair: 606

t: Amsterdam police said Wednesday that they have recovered stolen lithographs by the late U.S. pop artist Andy Warhol worth more than $1 million.

h: Police recovered 81 Andy Warhol lithographs.

RTE 3 Dev Set, Pair: 667

t: The British government has indicated its readiness to allow Argentine companies to take part in the development of oilfields in the Falkland islands' territorial waters.

h: The British government is ready to allow Argentine companies to participate in the development of oilfields.

RTE 4 Test Set, Pair: 862

t: Carole James is a newcomer in the leadership role of the NDP. She was elected in 2003 and aims to restore the party to power after they suffered an embarrassing defeat. In 2001, the NDP went from government to opposition with only two seats.

h: NDP won the 2001 election.

RTE 4 Test Set, Pair: 955

t: The largest search engine on the web, Google, receives over 200 million queries each day through its various services.

h: Google operates on the web.

Bibliography

Adams, Rod, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. 2007. Textual Entailment Through Extended Lexical Overlap and Lexico-semantic Matching. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, 119–124. Stroudsburg, PA, USA: Association for Computational Linguistics.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, 86–90. Stroudsburg, PA, USA: Association for Computational Linguistics.

Bar-Haim, R., I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Bar-Haim, Roy. 2010. Semantic Inference at the Lexical-Syntactic Level. Doctoral Dissertation, Bar-Ilan University.

Bar-Haim, Roy, Ido Dagan, Iddo Greental, Idan Szpektor, and Moshe Friedman. 2007a. Semantic Inference at the Lexical-syntactic Level. In *Proceedings of AAAI.*, 131–136.

Bar-Haim, Roy, Ido Dagan, Iddo Greental, Idan Szpektor, and Moshe Friedman. 2007b. Semantic Inference at the Lexical-Syntactic Level for Textual Entailment Recognition. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 131–136.

Beaver, David I. 2001. *Presupposition and Assertion in Dynamic Semantics*, volume 29. CSLI publications Stanford.

Beltagy, Islam, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep Semantics with Probabilistic Logical Form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM-2013)*.

Bentivogli, L., E. Cabrio, I. Dagan, D. Giampiccolo, M. L. Leggio, and B. Magnini. 2010a. Building Textual Entailment Specialized Data Sets: A Methodology for Isolating Linguistic Phenomena Relevant to Inference. In *Proceedings of LREC 2010*.

Bentivogli, L., I. Dagan, H. T. Dang, D. Giampiccolo, and B. Magnini. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of TAC*, volume 9, 14–24.

Bentivogli, Luisa, Peter Clark, Ido Dagan, Hoa T. Dang, and Danilo Giampiccolo. 2010b. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of TAC*.

Bentivogli, Luisa, Peter Clark, Ido Dagan, Hoa T. Dang, and Danilo Giampiccolo. 2011. The Seventh PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of TAC*.

Blackburn, Patrick, and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI.

Bos, J., S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier. 2004. Wide-Coverage Semantic Representations from a CCG Parser. In *Proceedings of the 20th international conference on Computational Linguistics*, 12–40.

Bos, J., and K. Markert. 2005. Recognising Textual Entailment with Logical Inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 628–635.

Bos, Johan. 2013. Is There a Place for Logic in Recognizing Textual Entailment. In *Perspectives on semantic representations for textual inference*, ed. Cleo Condoravi, Valeria de Paiva, and Annie Zaenen, volume 9 of *Linguistic Issues in Language Technology*.

Carlson, Gregory Norman. 1977. Reference to Kinds in English. Doctoral Dissertation, University of Massachusetts Amherst.

Celikyilmaz, Asli, Marcus Thint, and Zhiheng Huang. 2009. A Graph-based Semi-supervised Learning for Question-answering. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, 719–727. Stroudsburg, PA, USA: Association for Computational Linguistics.

Chierchia, Gennaro, and Sally McConnell-Ginet. 2000. *Meaning and Grammar: An Introduction to Semantics*. MIT press.

Chklovski, T., and P. Pantel. 2004. Verbocean: Mining the Web for Fine-grained Semantic Verb Relations. In *Proceedings of EMNLP*, volume 4, 33–40.

Cooper, Robin, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. 1996. *Using the Framework*. The Fracas Consortium.

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6)*. `http://tinyurl.com/gatebook`.

Curry, Haskell Brooks, and Robert Feys. 1958. *Combinatory Logic*. Number v. 1 in Combinatory Logic. North-Holland Publishing Company.

Dagan, Ido, and Oren Glickman. 2004. Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In *PASCAL WWorkshop on LLearning MMethods for TText Understanding and Mining*. Grenoble, France.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment* 3944:177–190.

Dagan, Ido, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool.

Dowty, David R., Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Dordrecht, Holland: D. Reidel.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. Mit Press.

Ferrández, Óscar, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrández, Matteo Negri, Rubén Izquierdo, David Tomás, Constantin Orasan, G'unter Neumann, Bernardo Magnini, and Jose Luis Vicedo. 2011. The QALL-ME Framework: A Specifiable-domain Multilingual Question Answering Architecture. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 1–12.

von Fintel, Kai. 1999. NPI Licensing, Strawson Entailment, and Context Dependency. *Journal of Semantics* 16:97–148.

Gamut, L.T.F. 1991. *Logic, Language, and Meaning*. University of Chicago Press.

Garoufi, Konstantina. 2007. Towards a Better Understanding of Applied Textual Entailment: Annotation and Evaluation of the rte-2 Ddataset. Master's thesis, Saarland University.

Garrette, Dan, Katrin Erk, and Raymond Mooney. 2013. A Formal Approach to Linking Logical Form and Vector-space Lexical Semantics. *Computing Meaning* .

Giampiccolo, D., H. T. Dang, B. Magnini, I. Dagan, and E. Cabrio. 2008. The Fourth PASCAL Recognising Textual Entailment Challenge. In *TAC 2008 Proceedings*.

Giampiccolo, Danilo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, 1–9. Stroudsburg, PA, USA: Association for Computational Linguistics.

Habash, Nizar, and Bonnie Dorr. 2003. A Categorial Variation Database for English. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, 17–23.

Harabagiu, Sanda, and Andrew Hickl. 2006. Methods for Using Textual Entailment in Open-domain Question Answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, 905–912. Stroudsburg, PA, USA: Association for Computational Linguistics.

Harabagiu, Sanda, Andrew Hickl, and Finley Lacatusu. 2007. Satisfying Information Needs with Multi-document Summaries. *Inf. Process. Manage.* 43:1619–1642.

Harabagiu, Sanda M., Marius A. Paşca, and Steven J. Maiorano. 2000. Experiments with Open-domain Textual Question Answering. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING '00, 292–298. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hickl, Andrew, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink, and Ying Shi. 2005. Recognizing Textual Entailment with LCC's Groundhog System. In *In Proc. of the Second PASCAL Challenges Workshop*.

Hobbs, Jerry R., Mark E. Stickel, Douglas E. Appelt, and Paul Martin. 1993. Interpretation as Abduction. *Artif. Intell.* 63:69–142.

Kamp, H., and U. Reyle. 1993. *From Discourse to Logic: Introduction to Model-Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42. Kluwer Academic Dordrecht,, The Netherlands.

Karttunen, Lauri. 1973. Presuppositions of Compound Sentences. *Linguistic inquiry* 169–193.

Katrenko, Sophia, and Assaf Toledo. 2011. A Comparison Study of Lexical and Syntactic Overlap for Textual Entailment Recognition. In *Proceedings of BNAIC 2011*.

Klein, Dan, and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, 423–430. Stroudsburg, PA, USA: ACL.

Kokke, Pepijn. 2013a. Lambdacalc. `https://github.com/pepijnkokke/LambdaCalc`.

Kokke, Pepijn. 2013b. Pelican. `https://github.com/pepijnkokke/pelican`.

Kouylekov, Milen, Yashar Mehdad, and Matteo Negri. 2011. Is it Worth Submitting this Run? Assess Your RTE System with a Good Sparring Partner. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*, 30–34. Association for Computational Linguistics.

Kruit, Benno. 2013. Using a First-Order Theorem Prover in the SemAnTE Framework. Bachelor thesis, Utrecht University.

Lin, D. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, 768–774.

Lin, D., and P. Pantel. 2001. DIRT@ SBT@ Discovery of Inference Rules from Text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 323–328.

Lotan, Amnon. 2012. A Syntax-based Rule-base for Textual Entailment and a Semantic Truth Value Annotator.

Ludlow, Peter, and Gabriel Segal. 2004. On a Unitary Semantical Analysis for Definite and Indefinite Descriptions. In *Descriptions and beyond*, ed. Marga Reimer and Anne Bezuidenhout. Oxford University Press.

MacCartney, B., and C. D. Manning. 2007. Natural Logic for Textual Inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 193–200.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. `http://www.aclweb.org/anthology/P/P14/P14-5010`.

Marelli, Marco, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 1–8. Dublin, Ireland: Association for Computational Linguistics and Dublin City University.

de Marneffe, Marie-catherine, and Christopher. D Manning. 2008. The Stanford Typed Dependencies Representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 1–8.

de Marneffe, Marie-catherine, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding Contradictions in Text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, 1039–1047. Colombus, Ohio: Association of Computational Linguistics.

McCune, William. 2010. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Mehdad, Yashar, and Bernardo Magnini. 2009. A Word Overlap Baseline for the Recognizing Textual Entailment Task .

Miller, G. A. 1995. WordNet: a Lexical Database for English. *Communications of the ACM* 38:39–41.

Montague, R. 1970. Universal Grammar. *Theoria* 36:373–398.

Partee, Barbara. 1975. Montague Grammar and Transformational Grammar. *Linguistic inquiry* 203–300.

Partee, Barbara. 1995. Lexical Semantics and Compositionality. *An invitation to cognitive science: Language* 1:311–360.

Potts, Christopher. 2014. Presupposition and Implicature. In *The handbook of contemporary semantic theory*, ed. Shalom Lappin and Chris Fox. Wiley-Blackwell, 2 edition.

Raina, Rajat, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust Textual Inference via Learning and Abductive Reasoning. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, 1099–1105. AAAI Press.

Riazanov, Alexandre, and Andrei Voronkov. 2002. The Design and Implementation of VAMPIRE. *AI Commun.* 15:91–110.

Richardson, Matthew, and Pedro Domingos. 2006. Markov Logic Networks. *Mach. Learn.* 62:107–136.

Romano, Lorenza, Milen Kouylekov, Idan Szpektor, and Ido Dagan. 2006. Investigating a Generic Paraphrase-based Approach for Relation Extraction. In *In Proceedings of EACL*, 409–416.

Roth, Dan, Mark Sammons, and V.G.Vinod Vydiswaran. 2009. A Framework for Entailed Relation Recognition. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, 57–60. Suntec, Singapore: Association for Computational Linguistics.

Sammons, Mark, V.G.Vinod Vydiswaran, and Dan Roth. 2010. Ask Not What Textual Entailment Can Do for You... In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1199–1208.

Sánchez Valencia, Victor. 1991. Studies on Natural Logic and Categorial Grammar. Doctoral Dissertation, University of Amsterdam.

Van der Sandt, Rob. 1992. Presupposition Projection as Anaphora Resolution. *Journal of semantics* 9:333–377.

Schlenker, Philippe. 2008. Be Articulate: A Pragmatic Theory of Presupposition Projection. *Theoretical Linguistics* 34:157–212.

Schlenker, Philippe. 2009. Local Contexts. *Semantics and pragmatics* 2:1–78.

Stern, Asher, and Ido Dagan. 2011. A Confidence Model for Syntactically-Motivated Entailment Proofs. In *Proceedings of RANLP 2011*.

Strawson, Peter Frederick. 1950. On Referring. *Mind* 59:320–344.

Van Strien, Gommaar. 2009. Semantic Annotations for Automatic Recognition of Textual Entailments.

Szabó, Zoltán Gendler. 2000. Descriptions and Uniqueness. *Philosophical Studies* 101:29–57.

Toledo, Assaf, Stavroula Alexandropoulou, Sophie Chesney, Robert Grimm, Pepijn Kokke, Benno Kruit, Kyriaki Neophytou, Antony Nguyen, and Yoad Winter. 2014a. A Proof-Based Annotation Platform of Textual Entailment. In *Proceedings of the 10th joint acl - iso workshop on interoperable semantic annotation (isa-10) in conjunction with the 9th international conference on language resources and evaluation (lrec)*, ed. Harry Bunt, 21–24. Reykjavik, Iceland.

Toledo, Assaf, Stavroula Alexandropoulou, Sophie Chesney, Robert Grimm, Pepijn Kokke, Benno Kruit, Kyriaki Neophytou, Antony Nguyen, and Yoad Winter. 2014b. Annotating by Proving using SemAnTE. In *Proceedings of the demonstrations at the 14th conference of the european chapter of the association for computational linguistics*, 77–80. Gothenburg, Sweden: Association for Computational Linguistics.

Toledo, Assaf, Stavroula Alexandropoulou, Sophia Katrenko, Heidi Klockmann, Pepijn Kokke, and Yoad Winter. 2013a. Semantic Annotation of Textual Entailment. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, 240–251. Potsdam, Germany: Association for Computational Linguistics.

Toledo, Assaf, Stavroula Alexandropoulou, Sophia Katrenko, Heidi Klockmann, Pepijn Kokke, and Yoad Winter. 2013b. Towards a Semantic Model for Textual Entailment Annotation. In *Perspectives on semantic representations for textual inference*, ed. Cleo Condoravi, Valeria de Paiva, and Annie Zaenen, volume 9 of *Linguistic Issues in Language Technology*.

Toledo, Assaf, Sophia Katrenko, Stavroula Alexandropoulou, Heidi Klockmann, Asher Stern, Ido Dagan, and Yoad Winter. 2012. Semantic Annotation for Textual Entailment Recognition. In *Proceedings of the Eleventh Mexican International Conference on Artificial Intelligence (MICAI)*.

Tonhauser, Judith, David Beaver, Craige Roberts, and Mandy Simons. 2013. Toward a Taxonomy of Projective Content. *Language* 89:66–109.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, 173–180. Stroudsburg, PA, USA: Association for Computational Linguistics.

Winter, Y. 2010. *Elements of Formal Semantics*. Unpublished ms., to appear with Edinburgh University Press. `http://www.phil.uu.nl/~yoad/efs/main.html`.

Summary in Dutch

Dit proefschrift presenteert een theoriegebaseerd paradigma om tekstuele entailment (logische gevolgtrekking) te onderzoeken. Dit paradigma gaat uit van de aanname dat implicatieherkenners nauwkeuriger kunnen worden gemaakt wanneer een expliciete taalkundige theorie in ieder geval een deel van de data verklaart waarvoor deze herkenners ontworpen zijn. Het proefschrift richt zich op de eerste stap om dit paradigma te realiseren: het ontwikkelen van een computationeel raamwerk dat een annotatiemethodologie mogelijk maakt waardoor entailmentdata die op zo'n theorie gebaseerd zijn kunnen worden verkregen.

De eerste vraag die onderzocht wordt luidt: op welke taalkundige fenomenen kan zo'n paradigma (doelmatig) toegepast worden? De hypothese is dat logische gevolgtrekkingen die afkomstig zijn van restrictieve modificatie (en de intersectieve en appositionele varianten daarvan), geschikt zijn, en voldoende algemeen voorkomen in natuurlijk taalgebruik, voor onderzoek van de voorgestelde methode.

Om deze hypothese te valideren werd gebruik gemaakt van de Recognizing Textual Entailment (RTE) corpora, de basis voor de evaluatie van entailmentherkenners in hedendaagse natuurlijke-taal verwerking. Deze corpora bestaan uit duizenden entailments die verschillende soorten logische gevolgtrekkingen representeren die regelmatig voorkomen in natuurlijk taalgebruik.

Het onderzoek is gebaseerd op een annotatieproject dat de frequentie onderzoekt van logische gevolgtrekkingen uit modificatiestructuren, gebruikmakend van de handmatige annotatie (twee uitvoerders) van de entailmentparen uit de RTE 1–4 datasets. De annoteerders markeerden alle gevallen van modificatie die een rol speelden in hun informele interpretatie van de entailmentrelatie tussen tekst en hypothese. Een kwantitatieve analyse van de annotaties bracht aan het licht dat in 80% van de positieve entailments in RTE1-4 logische gevolgtrekkingen veroorzaakt door modificatie gevonden werden. De geannoteerde datasets zijn vervolgens als corpus, getiteld *Semantic Annotation of Textual Entailment* (SemAnTE) 1.0, openbaar beschikbaar gesteld.

Tijdens het ontwikkelingsproces van SemAnTE 1.0 werd duidelijk dat het annoteren van logische gevolgtrekkingen en het verifiren van de nauwkeurigheid daarvan niet zonder problemen waren. Omdat de analyse informeel is en het om een vertaling gaat van een intutieve beoordeling van entailment naar een reeks concrete taalkundige verschijnselen, kan een annoteerder n of meerdere structuren niet opmerken, of onnauwkeurig annoteren. Onderlinge controle op annotaties (inter-annotator agreement, IAA) bleek geen bruikbaar instrument te zijn om zulke vergissingen op te sporen. In veel gevallen werden uiteenlopende annotaties gemaakt, omdat annoteerders verschillende legitieme paden volgden bij het herkennen van een entailmentrelatie. IAA-controles lieten in zulke gevallen een geringe overeenkomst zien, terwijl beide annotaties verdedigbaar waren.

Gebaseerd op de conclusies die getrokken zijn uit het werken met SemAnTE 1.0 is er een nieuwe methodologie ontwikkeld om entailments te annoteren, genaamd *Annotating-By-Proving* (ABP). Deze methodologie is gebaseerd op een model van het analyseren van logische gevolgtrekking gebaseerd op een standaard formeel-semantisch model. De hypothese is dat door de ABP-methodologie een annotatieplatform kan verifiren of de geannoteerde data verklaard worden door de semantische theorie: annoteerders krijgen feedback op hun annotaties. Om de bruikbaarheid van deze werkwijze te controleren beschrijft dit proefschrift de implementatie van een computationeel raamwerk dat gebaseerd is op de ABP-methodologie.

De voorgestelde aanpak ondersteunt de ontwikkeling van een getypeerd lexicon dat zowel modificatieverschijnselen als eenvoudige universele en existentile kwantificatie codeert. Aan woorden wordt een betekenis toegekend door hun verbinding met lexicale items, en aan complexe uitdrukkingen worden betekenissen toegekend die recursief gebaseerd zijn op de betekenissen van hun delen (compositionaliteit). Functie-applicatie in lambda-calculus vereenvoudigt de zinstermen die verkregen zijn voor tekst en hypothese. Het model definieert *Semantic Entailment* in een gegeven paar door het waarheidsconditionele criterium toe te passen op de sententile termen van tekst en hypothese. De taalkundige verschijnselen worden geanalyseerd met behulp van settheoretische concepten. Het model veronderstelt eenvoudige representaties die toegepast worden op entailmentdata waarbij standaard tree-taggingconventies worden aangehouden. Verondersteld wordt dat compositionaliteit van centraal belang is voor de leervermogen van het model. Deze aanpak kan beginnen met een theorie die een kleine set taalkundige verschijnselen modelleert en die vervolgens uitgebreid kan worden met complexere syntactische constructies en semantische logische gevolgtrekkingen.

Er wordt een gemplementeerd annotatieplatform beschreven dat automatisch een entailmentpaar verwerkt door gebruik te maken van de volgende componenten: (1) een syntactische parser en een part-of-speech-tagger om een parseringsboom met syntactische categorielabels te verkrijgen, (2) een heuristiek om part-of-speech-tags te gebruiken om lexicale tokens te verbinden met het lexicon van het systeem, (3) een controlesysteem voor consistentie van de typering, om te valideren dat functie-applicatie mogelijk is voor de parseringsstructuren van tekst en hypothese, (4) een bta-reducer om de zinstermen van tekst en hypothese te vereenvoudigen, (5) een orde-verlagend algoritme om de termen in hogere-ordelogica te verlagen naar eerste-ordelogica, en (6) een bewijsalgoritme ontworpen om, wanneer mogelijk, een formeel bewijs te genereren dat de term, berekend voor de tekst, logisch leidt tot de term van de hypothese. Menselijke annoteerders markeren de lexicale relaties en corrigeren de parsing en de vergissingen in annotatie die het resultaat zijn van onvolkomenheden in de automatische instrumenten gebruikt voor componenten (1) en (2) zoals hierboven genoemd. De door het platform gebruikte instrumenten en technieken verbinden de syntactisch-semantische tagging van parseringsbomen in natuurlijke tekst, modeltheorie, en bewijstheorie.

Een cruciale overweging bij het ontwikkelen van het platform was om het solide te maken met betrekking tot de semantische theorie en redelijk gevoelig, zelfs als dat ten koste ging van volledigheid. Met betrekking tot elk paar zinnen laat het systeem zien dat het bewijs dat het genereert een betrouwbare indicatie is dat een theoretisch bewijs afgeleid kan worden binnen het toegepaste semantische model. Dit maakte het mogelijk om ABP te verwezenlijken: een entailing paar kan alleen als goed geannoteerd beoordeeld worden als de annotaties kunnen dienen om een bewijs te genereren tussen de representaties van de tekst en de hypothese. Een uitbreiding van deze methodologie omvat ook non-entailing (negatieve) paren: een non-entailing paar wordt alleen als goed geannoteerd beschouwd als het platform niet in staat is om een bewijs te genereren op basis van de annotaties; terwijl het aan de andere kant in staat is om een bewijs te genereren voor een minimaal verschillend entailing paar dat zo gelijk mogelijk geannoteerd is. Het minimale contrast tussen een entailing paar en zijn bewijs aan de ene kant, en een non-entailing paar, en gebrek aan bewijs aan de andere kant, rechtvaardigt de conclusie dat het gebrek aan bewijs voor het non-entailing voorbeeld te wijten is aan semantische overwegingen, en niet te wijten is aan een vergissing in de annotatie of aan onvolledigheid van het systeem.

Het platform diende om de toepasbaarheid van het semantische model op entailmentdata te evalueren. Om dit doel te bereiken is een nieuw corpus van geannoteerde entailments gecreerd, genaamd SemAnTE 2.0. Het corpus bestaat uit twee delen: (1) entailing (positieve) paren die gecreerd zijn door RTE-voorbeelden te vereenvoudigen, en (2) tweetallen van nieuwe minimaal contrasterende positieve en negatieve paren die voldoende zijn om contrasterende entailmentbeoordelingen te kunnen uitlokken binnen het semantische model. Door deze methodologie worden alle entailmentbeoordelingen in SemAnTE 2.0 verklaard door het semantische model. De dataset bestaat uit 600 paren met een positief-negatief ratio van 2:1 en is vrij beschikbaar op het web.[1]

Door de implementatie van de Annotating-By-Proving methode te demonstreren, ondersteunt dit proefschrift de aanname dat de standaard formeel-semantische theorie uitvoerbare computationele modellen van tekstuele entailment kan genereren. Relatief eenvoudige representaties gecodeerd in seman-

[1] http://logiccommonsense.wp.hum.uu.nl/resources

tische modellen hebben voldoende verklarend vermogen om een ruime verscheidenheid aan logische gevolgtrekkingen in natuurlijke taal te verklaren. De beschikbaarheid van een expliciete taalkundige theorie die de data, gecreerd in dit paradigma, verklaart, houdt de belofte in dat nog beter werkende entailmentherkenners ontwikkeld kunnen worden.

Curriculum Vitae

Assaf Toledo was born on September 17, 1980, in Netanya, Israel. He studied Mathematics at Tel Aviv University between 2000 and 2005, graduating with a Bachelor's degree. In 2008 he joined the Master's program in Linguistics at Hebrew University of Jerusalem, and graduated cum laude in 2010. In 2010 he was accepted to the PhD program at UiL OTS, Utrecht University, the Netherlands. This dissertation is the result of his PhD research.