Flat but not shallow

Towards flatter representations in deep semantic parsing for precise and feasible inferencing

Published by
LOT
Trans 10
3512 JK Utrecht
the Netherlands

Phone: +31 30 253 6006
Fax: +31 30 253 6000
e-mail: lot@let.uu.nl
http://wwwlot.let.uu.nl

Cover illustration: Earthrise seen for the first time by humans eyes. Photo taken by the crew of Apollo 8, 24 December 1968. Made freely available by NASA.
http://www.hq.nasa.gov/office/pao/History/alsj/a410/AS8-13-2329HR.jpg

ISBN: 978-90-78328-80-3
NUR: 616

This dissertation is typeset using LaTeX.

Flat but not shallow

Towards flatter representations in deep semantic parsing for precise and feasible inferencing

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van de Rector Magnificus prof. mr. P.F. van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op Woensdag 18 maart 2009
klokke 16.15 uur

door

Hilletje Gezina Bouwke Reckman

geboren te Groningen
in 1978

Promotiecommissie

Promotores:	Prof. dr. V.J.J.P. van Heuven Prof. dr. J.H.M. Schonk
Co-promotor:	Dr. C.L.J.M. Cremers
Referent:	Prof. dr. F. de Jong (Universiteit Twente)
Overige leden:	Dr. B. Arsenijević (Universiteit van Amsterdam) Prof. dr. L.L. Cheng Prof. dr. G.A.M. Kempen (Max Planck Instituut voor Psycholinguistiek) Dr. A. van der Wouden

The research reported here was conducted in the context of the ToKeN-project 'Narrator', funded by the Dutch Organization for Scientific Research (NWO).

To my parents

Contents

Acknowledgements

This thesis is a result of the Narrator project, made possible by NWO. I would like to thank NWO and the Narrator group, in particular Regina Overberg, Pieter Toussaint, Leo Wolf, Henk Herman Nap, Eduard Hoenkamp, and Gerard Kempen, the other 'kikkers', and Christiane Klöditz from NWO.

Above all, I would like to thank those whose names in good Leiden tradition are conspicuously absent form this list and the ones below, but without whom this thesis would not have been there.

At LUCL I want to thank Boban Arsenijević, Anita Auer, Linda Badan, Sandra Barasa, Birgit Bexten, Sylvia Blaho, Hans Broekhuis, Martine Bruil, Leston Buell, Lisa Cheng, Liesbeth De Clerck, Camelia Constantinescu, Jeroen van Craenenbroeck, Jenny Doetjes, Marius Doornenbal, Marion Elenbaas, Noureddine Elouazizi, Barbara Floris, Egbert Fortuin, Thea Gagnidze, Veronique van Gelderen, Dafna Graf, Stella Gryllia, Gea Hakker, Margarita Gulian, Sita ter Haar, Anne-Christie Hellenthal, Allison Kirk, Annemiek Hammer, Willemijn Heeren, Roland Hemmauer, Pepijn Hendriks, Maarten Hijzelendoorn, Juliette Huber, Mélanie Jouitteau, Alwin Kloekhorst, Elisabeth Koier, Jan Kooij, Marjo van Koppen, Maarten Kossmann, František Kratochvíl, Leontine Kremers, Guus Kroonen, Nancy Kula, Nana Kusuma, Melissa Kwee, Stephen Laker, Frank Landsbergen, Maarten van Leeuwen, Claartje Levelt, Boya Li, Kathrin Linke, Anikó Lipták, Sara Lusini, Kristin Meier, Alice Middag, Ineke van der Meulen, Maarten Mous, Karlijn Navest, Victoria Nyst, Marc van Oostendorp, Jos Pacilly, Michaël Peyrot, Mika Poß, Tijmen Pronk, Felix Rau, Chris Reintges, Kristina Riedel, Johan Rooryck, Martin Salzmann, Graziano Savá, Thilo Schadeberg, Jos Schaeken, Niels Schiller, Franziska Scholz, Erik Schoorlemmer, Joanna Sio, Arlette Sjerp, Káča Součková, Sander Steeman, Robin Straaijer, Rint Sybesma, Amir Tauber, Tanja Temmerman, Kalinka Timmer, Elena Tribushinina, Rada Trnavac, Assimakis Tseronis, Marina Tzakosta, Michiel de Vaan, Marijn van 't Veer, Rinus Verdonschot, Arie Verhagen, Margreet Verra, Luis Vicente, Rebecca Voll, Mark de Vos, Jenneke van der Wal, Jeroen van de Weijer, Jurriaan Witteman, Leo Wong, and Ton van der Wouden. Although not members of LUCL, to me Merijn de Dreu and Asad Jaber also belong in this list. You, the inhabitants of 1166 and surrounding buildings, were a great community to be part of. Special thanks to Thea for helping me integrate into the PhD crowd before I even started. And a big thank you to Jeroen, Gea, Margreet, Alice and Jos for keeping the institute up and running.

Sitting alone in the office the whole day is not something I enjoy very much. Therefore I am especially grateful for the officemates I had over the years: Boban (my long-term officemate and older brother in linguistics), Noureddine, Leo, Stella, Jeroen, and my present officemate Rinus. Lately my officemates have not been there very much, but fortunately having Maarten next door pretty much made up for that. Thanks Maarten,

for all your help. I am also thankful to Erik for regularly dropping by at my office with his cheerful presence and the latest gossip, to Birgit for giving me tea every time I dropped by at her office, and to Mika for good stories and common linguistic interests and trips. And of course thanks to everyone who picked me up for lunch.

I also would like to express my thanks to the organizers of LUSH; Berit, Jakub, Jenny, Marieke, Káča, Bert, Evangelia, Matteo, and all the others involved. It was good to have regular semantics talks to go to.

In the course of my PhD I attended lots of LOT schools and one of the Egg schools. I very much appreciated this opportunity. Courses that I found particularly useful and/or inspiring were those taught by Antal van den Bosch, Antonella Sorace, Miranda van Turenhout, Simon Garrod, Rick Nouwen, Marcus Egg, James Pustejovsky, Anatol Stefanowitsch, and Luc Steels. I am also very happy about having met so many other students at these schools. Here I stand no chance of providing a complete list, but Rafał, Kasia, Asia, Olga, Irene, Wieneke, Maren, Nino, Øystein, Jutta, Nataša, Veronika, Ingrid, Ming, Janneke, Diana, Roberta, Eleonora, Anne, Hedde and Robert should certainly be in there.

Taking courses is great, but so is teaching. I was lucky to get the chance to teach, and to have students who enthusiastically participated in the courses.

I express my gratitude as well to the audiences of the talks I gave, for their attention and comments, and I thank LUF for sponsoring my participation in ICoS-5.

Since the home environment is as important as the work environment, I'm saying thanks to the neighbors from the PhD ghetto, in particular Graciana, Eva, Frank, David, Andreas, Kristo, Hana, Christian, Otto, Rob, Esther, Orion, and Percy. And thanks to Martijn and Moira for helping us take over their apartment.

Others that I could always count on to give me a needed break from linguistics are the astronomers; Dominic, Demerese, Sergio, Bob, Helen, Stijn, Remco, Sarah, Liesbeth, Ned, Niruj, and many others.

I thank Diana, Gabriëlle, Zuzana, Mathieu, and Selina for their friendship, and Ayman for making me more aware of my ambitions, helping me realize that a PhD was what I wanted to go for.

In the end, I guess, I owe this success largely to my family who I can always count on and who have always been convinced of my intellectual capabilities and encouraged me to use them.

And finally Simon, thanks for your continuous love and support and for always believing in me, and also for telling me lots about astronomy and for making me rediscover sports, which greatly contributed to me surviving my desk job. Meeting you has upgraded my life considerably in many ways, and I'm looking forward to a lot more time with you.

Introduction

Simulating natural language understanding on the computer is a great challenge. Advancements in this field will not only provide valuable insights into how human language works, but also create possibilities for applications with a far reaching impact on the ways in which we store and retrieve data. (Imagine having the kind of cooperative speaking computer they use in Star Trek.)

A way to approach this challenge is by building systems that translate natural language sentences into logical propositions. Such semantic representations are referred to as logical form. Provers have been implemented for several logics, allowing for automated reasoning to be performed on information presented in these logics. First-order logic is popular among those who want to do 'deep' semantics; full propositional analysis including quantification. Reasoning with first-order logic is however computationally demanding and therefore too slow for many purposes. As a result there is a trend toward using formalisms of 'shallow' semantics, which are easier to process, but unfortunately less expressive. Many aspects of natural language meaning cannot be adequately represented in these. In particular, quantifiers tend to be ignored.

Logical form is considered to be a level of grammatical representation at which semantic consequences can be computed (Higginbotham, 1985). Semantic consequences are called entailments. The computing of such entailments is called inference. Entailment will be a central notion in this thesis. If the truth of a sentence *A* necessarily makes a sentence *B* true as well, we say that *A* entails *B*. We will see more formal definitions of entailment in chapter one.

In this dissertation a format of logical form is developed that is easier to process than first-order logic, while actually being able to express more linguistic meanings. It was especially designed to make the computing of many kinds of entailments as easy as possible. The entailment patterns induced by a wide range of natural language determiners/quantifiers can be captured. These logical forms are rather flat in structure. Here, 'flat' means that the information is not arranged hierarchically, which keeps the structure simple. The logical forms are also rich in information, incorporating, for example, modern insights in the semantics of eventualities in words of different grammatical categories. All together this makes the representations flat but not shallow; manageable in processing but rich in semantic information relevant to entailment.

The research presented in this thesis was conducted in the context of the Narrator project. This project, which is part of the NWO ToKeN program, aimed at developing an information system that provides retrieval on essentially free narrative text. The work in this thesis focuses on automated semantic analysis obtained through deep parsing. The objective is to develop meaning representations that are optimized for retrieval by means of a form of automated reasoning.

The approach taken relies on the hypothesis that explicit knowledge of language can be assembled, formalized and exploited up to full semantic interpretation, and is considered complementary to statistically based and sub-symbolic strategies to language processing.

Computational linguistics started off with linguists building systems that were based on hand-coded grammar rules. This method was soon found to have several disadvantages. The hand-coding was very labor-intensive, and grammars and lexicons were never complete. In addition, there were problems of huge and often spurious ambiguity. Hence, such attempts resulted in systems that left much to wish for in terms of efficiency and robustness. The development of statistical approaches led to fast progress on these points. Statistics-based methods for tagging, chunking, shallow, and deeper syntactic analysis are by now quite well consolidated and have proven to provide a solid basis for natural language analysis. Recently, however, there is a growing consensus that to go beyond this basis and simulate a more fine-grained understanding of language, shallow methods need to be combined with so called 'deep' analysis. For instance, performance in the recent TREC Question Answering tracks has been reported to show that inferencing substantially improves response relevance and accuracy (Dang et al., 2007; Voorhees and Harman, 2005). The growing interest in the Semantic Web is a related development. Its goal is to develop a common semantic annotation scheme for web data (Berners-Lee et al., 2001). Ideally it should be possible for the annotation to be performed automatically. At the same time several more linguistically informed search engines, which want to compete with Google, are emerging. The need for a closer approximation of natural language understanding reflected in these developments is one of the main motivations for the course of research pursued in this thesis.

Attempts to build systems that automatically analyze or produce natural language have at least two major purposes. One purpose is to increase understanding about how language works in humans by means of modeling. The other is to develop all kinds of useful applications. Often, the same research can be relevant to both goals, but sometimes they require different priorities. Consider, for example, information retrieval systems that help us process the overload of information we are confronted with. These are among the most urgent applications to be developed. They are on the one hand required to process text much faster than a human reader can, but on the other hand they do not necessarily need to replicate the full human understanding of a text in order to be useful. This means that for some tasks it can be useful to rely only on the specific strengths of the computer, rather than trying to replicate the way a human would approach it.

The project of this thesis is application oriented in its aims. Still, the research carried out is rather fundamental in nature. To the extent that it is available, psycholinguistic research is considered an important source of information and inspiration. The human brain proves so far to be the best natural language processor that exists, so any information about how this process works is relevant in principle. Unfortunately very little is known so far about how meaning is processed in the brain, and those psycholinguistic insights that do exist are not always straightforward to implement. This is true not least because the way that computers can approach language is fundamentally different from the way humans do. Computer systems typically have no access to

meaning at all. For humans, understanding the meaning of words and utterances is an essential part of language acquisition. No human child will acquire a language just by being exposed to a corpus, when there is no way to find out what the utterances in the corpus could possibly be about. An exception to the usual computer systems are robots such as the ones developed by Steels and his group, which develop their language by playing language games. They can be more like humans, because they are aware of their surroundings and learn by interaction (Steels and Kaplan, 2001).

The success of statistic parsers shows that it turns out to be possible to learn a lot about syntactic structure, starting out with minimal assumptions, merely by looking at the distribution of word forms, and without any access to meaning. To some extent, computers can work around the meaning problem by comparing lots of data. After all, if semantics is indeed (as computational semanticists assume) compositional, then it follows in a predictable way from words and structures. Even though it is not possible for a system to know what an utterance is about, it might be able to figure out if it is likely to be about the same thing as some other utterance. However, this has its limits, because more knowledge is needed than can be extracted. It is hardly possible to find systematic and reliable data to learn about different ways of paraphrasing. In order to compute meaning from words and structures, one needs to know the meaning of those words and structures. Especially the meanings of function words are crucial. For automated reasoning it is, for example, important to know the monotonicity properties of quantifiers.

Achieving high precision in advanced tasks, such as question answering, or open domain textual retrieval that goes beyond keyword search, requires a semantic level of analysis. The semantic representations need to be such that they are suitable input for automated reasoning algorithms. The problem with meaning is that it is, so far, not quite clear how best to represent it. Meaning is much harder to grasp than syntactic structure. For syntactic analysis the words need to be organized in hierarchical structures. This is potentially also very difficult, but it has become clear that for practical applications surface oriented syntactic analysis is quite successful. As a matter of fact, for computing deep semantics, a syntactic basis of this type also appears to be sufficient. First-order logic is often used for deep semantic analysis, but is not an ideal solution, both because it is not a perfect match in terms of what can be expressed, especially in the domain of quantifiers, and because reasoning with first-order logic is still very hard. And even if the general format were decided on, there are still many issues to be resolved. We will see several examples throughout the chapters, concerning issues like events, participant roles, degrees and quantification.

Assembling the knowledge needed for fine-grained semantic analysis, in the form of lexicons, ontologies and the like is a huge task. This does not mean that it is not feasible — think of the investments that are being made in conventional dictionaries. It only needs to be worthwhile. And for computational semantics resources it will be worthwhile as soon as there are well-consolidated, useful and satisfying methods of semantic representation. The development of such methods is what this thesis is intended to contribute to. The main criterion will always be that the representations have to support wanted entailments while blocking unwanted ones.

This research concentrates on the semantic component of the Delilah parser. This is

the first and so far the only compositional semantic parser for the Dutch language. It is an operational model of how logical form is computed.

The thesis consists of four chapters. The first chapter discusses the Narrator project as a retrieval task. Also, the notion of entailment and the problems of automated inference are discussed in more detail, in the context of the Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2006). It serves to set the background and present the current state of research in inference and retrieval. The rest of the thesis sets out to contribute to a way of improving the logical inference track to retrieval and related tasks, through the development of better suited semantic representations.

Chapter two introduces the Delilah parser and generator, which is designed to compute logical representations of texts. It is a strongly lexicalist system, in which the lexicon consists of typed feature structures. The basis of the system is a categorial grammar with rigid types and multiple modes of composition. The main focus of the Delilah system is precise semantic analysis. In the chapter I explain how the semantic analyses are derived, starting out from the grammar and the lexicon. I discuss some problems that still exist in the system, such as some syntactic gaps, matters of quantifier scope, and robustness issues.

Chapter three discusses the implementation of event semantics. This is a first step towards making the representations flatter and giving them more handles for inference. I discuss how event semantics was implemented for the different types of verbs, but also for nominalizations. The approach is extended to states expressed by adjectives and abstract nouns. The results are shown to be relevant for light verb constructions. The choices that were made are discussed in the context of inference. The representations are designed to make the correct entailments follow in a straightforward manner.

The fourth chapter then explores a way of representation that is flatter than first-order logic, while incorporating higher-order quantifiers: Flat Logical Form (FLF), designed to allow for improved inference-based retrieval. FLF is a conjunction of predicates over variables, in which the quantifier that binds the variable, the quantifiers and other operators that this quantifier is dependent on, and the entailment direction for the predicate resulting from these two factors are coded on the variable. The resulting representations are more informative than standard logical representations and at the same time easy to process.

Both event semantics and FLF thus contribute to making the representations flatter. The representations are now less hierarchical and contain less embedding and thus have a simpler syntax that makes more parts of the representation available at a higher level. FLF takes the form of a conjunction, which is a maximally flat structure, because hierarchy is irrelevant, as the results of all possible ways of assigning hierarchical structure are equivalent. Some relations between the conjuncts do need to be taken into account in FLF, but it is always locally decidable which ones. For example, if a conjunct is dependent on another one, this is marked on the conjunct itself. In other systems, the choice of flat representations that ease processing often results in shallower semantics, that ignores, for example, quantification. The flat FLF representations are, however, not shallow. They are at least as rich in information as the event semantic representations in the traditional format.

Whereas the long term goal in this research is to contribute to an application like Narrator, the immediate goal in this thesis is to develop ways of semantic representation in Delilah that are more suitable for automated inference in terms of both detail and feasibility. Implementation is discussed to show how the representations argued for can be computed.

Chapter 1

The Narrator project, retrieval, and inference

Narrator is a project in the NWO ToKeN[1] program, in the health-care domain. It's goal is to develop the Narrator system, a system accessible via internet that discloses narrative texts relating the experiences of breastcancer patients. Narrator will help breast cancer patients to find matching stories of others on the internet.

The Delilah parser, protagonist of the other chapters in this thesis, is to be a central part of the language technology used in Narrator. An important ambition in the development of Delilah is that the semantic representations that are its output will facilitate automated inference, for example for the retrieval of information and/or documents. In this chapter I will discuss this against the background of the Narrator project. Building on the strengths of Delilah, Narrator pursues a method of retrieval that involves logical inference on semantic representations.

In the first section I give some information on the Narrator project and system. And in the second section I give a brief sketch of the field of information retrieval and how the perspectives for Narrator fit in there. Together these sections will provide a somewhat more concrete picture of the kind of retrieval task it is about. It is explained how Delilah and a retrieval tool that works on its representations are envisaged to work together with other components. The next section discusses the difficulty of automated inference, which is the road to retrieval that we focus on, and some work that has been done in the context of the Pascal Recognizing Textual Entailment (RTE) challenge. This section about inference is of the most direct relevance to the work described in the other chapters. The rest is background, sketching the long term perspectives of such efforts.

The Narrator system has not yet been built. Overberg (forthcoming) explored the role Narrator could play in the sharing of experiences between patients. The present work focuses on detailed semantic representation to be used in the logical track to inference, which in turn can be employed in retrieval tasks where precision is prioritized. In this chapter I explain how and why the approach taken is expected to contribute to retrieval in a system like Narrator.

[1] ToKeN stands for *Toegankelijkheid en Kennisontsluiting in Nederland* 'Accessability and Knowledge disclosure in the Netherlands'

1.1 The Narrator project and system

This section describes the Narrator project in some more detail. The Narrator project started out as a joint project of the Clinical Informatics group of the Leiden University Medical Centre (LUMC), the J.F Schouten school for User-system Interaction of the Technical University Eindhoven (IPO institute), and the Leiden University Centre for Linguistics (LUCL). Later, the Nijmegen Institute for Cognition and Information (NICI) also joined. It serves as a larger frame to give direction to the different sub-projects, among which is the work of this thesis.

The project aims at the development of a natural language dialogue system that discloses personal narratives to facilitate patients (and their relatives) in finding relevant experiences of their fellow patients. The system should take into account their personal profiles and information need (Wolf et al., 2006). The particular case studied in the project is the development of such a system for female breast cancer patients. This is a group of patients for which there was evidence that they could benefit from such an information system, especially in the period after treatment.

It was decided that the system will be a web-based application, so that access is anonymous, and available from any place with an internet connection, e.g. home or hospital. The required level of detailed analysis to support a patient in finding appropriate illness stories in the diverse set of available stories is to be provided by natural language analysis techniques that facilitate content-based retrieval.

There already are websites available on which breast cancer patient can share their experiences. However, none of the websites implements search facilities based on either the content of an illness story, or the personal features of the author. A minority of the websites offer information about the illness stories, such as the author's genuineness and editor's review (Overberg et al., 2006).

Figure 1.1 from Wolf et al. (2006) gives an overview of the functional design of the Narrator system.

The basic component of the system is of course a database with narratives by breast cancer patients. These are initially collected from various sources by the developers of the system. When the system is up and running users can also send in their own stories. Narratives are free text and as a consequence they will vary a lot in length and style, and typically contain some imperfections, such as unfinished sentences or errors in spelling and grammar. Some obstacles to parsing, such as typos, can be repaired. Other problems are solved by making the parser more robust. For example, for unfinished sentences it is preferred to give a partial parse, rather than to complete the sentence by hand before parsing, which would compromise the authenticity of the document.

The narratives are to be parsed by the Delilah parser resulting in semantic representations of the text. Parsing happens off-line and the semantic representations are stored alongside the texts. This allows in principle for manual corrections or adaptations to the representations if needed. Moreover, words that occur in the text, but not in the lexicon, can be added to the lexicon and the text can be re-parsed for a more complete representation.

For retrieval, a combination of logic-based retrieval tools and statistics-based retrieval tools, such as Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Hoenkamp,

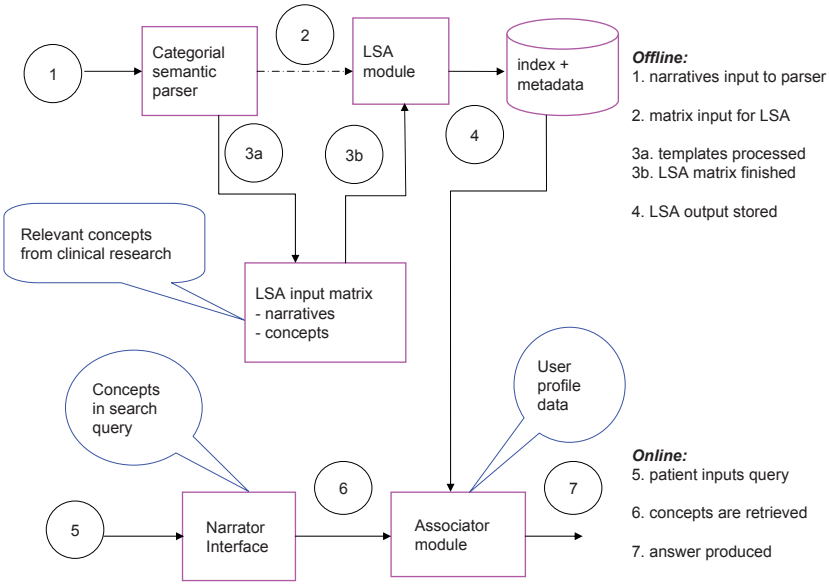


Figure 1.1 — The Narrator system in offline and online mode

2003; Landauer et al., 1998) was conceptualized. The integration of these is discussed in section 1.2.3.

At least the logic based retrieval is aided by the use of ontologies for extra information. Quite elaborate medical ontologies are available, but their role in Narrator is rather limited as the focus of the narratives is in general not medical. Instead, narratives tend to deal with issues like coping with emotional aspects and the impact of cancer on daily life (Overberg et al., 2007). These are also the topics that users are most interested in. Retrieval therefore also benefits from more general ontologies, for example of the WordNet type (Miller et al., 2004). For Dutch, the most promising database of lexical semantic information is Cornetto (Vossen et al., 2007)(see also 3.1.3).

Key concepts relevant to the topics users are most interested in can be given extra weight in the LSA clustering . Good candidates are, for example, emotion terms. Stories, or story fragments can also be annotated with terms that reflect their main topics, for direct search.

Another source of information is meta-data about the author, such as age, family situation, type of treatment received, presence of metastases, and time since diagnosis. This information can be added to the text as logical propositions to provide background information that is taken into account in retrieval, in addition, it can be directly used as search criteria. Potential users show interest in such search options (Overberg et al., 2007).

In the user interface, the user enters a query. Apart from the option of a detailed

search in natural language, it is probably a good idea to have some 'ready made' search options, with which users can search for stories with the most popular features for content and author information. This will enable users to familiarize themselves with the system and the kind of stories in it, before starting to formulate their own queries for a more specific search.

An associator module acts as a kind of mediator between the off-line mode (preparing the narratives) and the online mode (processing the query and finding a match). It uses structures generated in the off-line process as well as query information from the user interface to produce an answer. Information from previous online sessions (in the form of profiles) is kept to further assist in the search and retrieval process.

1.2 Information Retrieval

Information retrieval is concerned with locating information that will satisfy a user's information need. The present section is about the retrieval of written text, in tasks like document retrieval and question answering. The typical traditional text retrieval system is the library catalogue that helps the user find books that fulfill certain criteria. A type of document retrieval system that is very commonly used nowadays are search engines on the internet.

1.2.1 Different types of information retrieval

The Text REtrieval Conference (TREC) can be considered to represent the state of the art in Information Retrieval. It is a series of workshops designed to foster research on technologies for information retrieval. The Cross-Language Evaluation Forum (CLEF) is its multilingual counterpart. Each year they run a number of different tracks. The tracks differ in themes and tasks. As Voorhees (2007) puts it:

> The tracks serve several purposes. First, tracks act as incubators for new research areas: the first running of a track often defines what the problem really is, and a track creates the necessary infrastructure (test collections, evaluation methodology, etc.) to support research on its task. The tracks also demonstrate the robustness of core retrieval technology in that the same techniques are frequently appropriate for a variety of tasks. Finally, the tracks make TREC attractive to a broader community by providing tasks that match the research interests of more groups.

A text-collection is made available for each track. Test collections consist of three parts: a set of documents, a set of information needs (called topics in TREC), and relevance judgments, an indication of which documents should be retrieved in response to which topics. The result of a retrieval system executing a task on a test collection is called a run.

TREC distinguishes between a statement of information need (the topic) and the data structure that is actually given to a retrieval system (the query). Participants are free to

use any method they wish to create queries from the topic statements, whether automatic or manual.

Over the years there have been two main trends in the development of the text collections. They moved from smaller to larger sets of documents, and from news wire texts to a broader spectrum (such as recordings of speech, web pages, scientific documents, blog posts, email messages, and business documents).

Different types of tasks can be distinguished. The most typical one is the ad hoc task, in which the system cannot anticipate the particular topic that will be investigated. A retrieval system's response to an ad hoc search is generally an ordered list of documents sorted such that the documents that the system believes are more likely to satisfy the information need are ranked before the documents it believes are less likely to satisfy the need. In a categorization task, on the other hand, the system is responsible for assigning a document to one or more categories from among a given set of categories. Deciding whether a given mail message is spam is one example of a categorization task. The 2007 blog track contained a polarity task, in which opinions were determined to be 'pro', 'con' or mixed. This is a second example. A task can require to return an entire document or precisely the answer to a question. Different levels of granularity in between are also possible.

The relevance of a document to a topic is judged by trained annotators. In order to avoid having to annotate each document for each topic in a large collection of documents, a technique called pooling is used. In pooling, the top results from a set of runs are combined to form the pool and only those documents in the pool are judged. Runs are subsequently evaluated assuming that all unpooled (and hence unjudged) documents are not relevant.

Important evaluation measures are precision and recall at different cut-off levels. Precision is the proportion of retrieved documents that are relevant (number-retrieved-and-relevant/number-retrieved). Recall is the proportion of relevant documents that are retrieved (number-retrieved-and-relevant/number-relevant). A cut-off level is the number of retrieved documents that is taken into consideration, starting from the highest ranked one. For example, with a cut-off level of ten, the top ten documents in the ranked list are the ones that count for the evaluation. An imposed cut-off level is needed because a ranking task aims to rank documents from most relevant to least relevant, rather than categorize them as either relevant or irrelevant.

Many of the tasks in TREC are variants of document retrieval through keyword search. It turns out that in this type of task the best results are obtained with very limited use of linguistic analysis. This is also reflected in the success of the internet search engine Google. There are three main strategies (which can be combined) (Kuropka, 2004). Set-theoretic models, for example the standard Boolean model, represent documents as sets of words or phrases. Similarities are usually derived from set-theoretic operations on those sets. Algebraic models, such as vector space models, represent documents and queries usually as vectors, matrices or tuples. The similarity of the query vector and document vector is represented as a scalar value. Probabilistic models, such as Bayesian models, treat the process of document retrieval as a probabilistic inference. Similarities are computed as probabilities that a document is relevant for a given query. Additional

tools that have proven important are stemming, phrasing (recognizing collocations as a single unit), and the use of weighting schemes.

In the Question Answering track on the other hand, more linguistically informed methods, such as parsing and semantic role labeling are used in many systems. For retrieval from large corpora a layered approach is common (Jurafsky and Martin, 2000). Potentially relevant passages are first selected by general retrieval methods of the types mentioned above and only those passages are parsed for a more precise evaluation. In TREC 2007, for example, the linguistically principled Pronto QA system (Bos et al., 2007) scores above average. (The inferencing system used in Pronto also participated in some of the RTE challenge mentioned below.) The task of question answering is ultimately AI-complete. It is therefore not surprising that it benefits from detailed linguistic analysis to simulate understanding of the question and passages containing possible answers, as well as automated reasoning techniques.

Retrieving documents on the basis of keyword search is a much more coarse-grained task. It turns out that the weakest link in the retrieval procedure is often the quality of the query. Systems attempt to remedy this problem with techniques like query expansion.

It is clear that keyword search has its limitations. Lately new search engines have appeared, making use of more linguistic analysis, and aiming to compete with Google. Examples are Powerset and Hakia. The developers of each of these two point out that if users enter phrases or full questions as queries, they are most likely to profit from the strength of these engines and obtain better results than with Google. It is claimed that this is where linguistic analysis makes a difference. Powerset once used the example of the difference between *books by children* and *books for children*. Google will consider the difference in preposition irrelevant, making it difficult to find the rarer documents about the former between the more abundant documents about the latter.

Sometimes when things are hard to find with Google, clever use of double quotation marks can help. I remember once friends of mine wanted to know what the world's biggest harbor was, but couldn't find it when they searched the internet. I found an answer in the end by giving as a query something like *"is the world's biggest harbour"*. To get more results it is natural to try variations, such as *"is the largest harbor in the world"*. This method of thinking what might be the literal wording of part of the information you are looking for is a way of trying to circumvent Google's limitations and in general the limitations of keyword search and search algorithms that are insensitive to linguistic structure.

1.2.2 Information retrieval in Narrator

What kind of retrieval task fits the Narrator system? Here we have a manageable amount of narratives, and hence we have the opportunity to parse all narratives and allow for fine-grained search with queries in the form of phrases or questions, possibly next to the option of keyword search. Since users might be looking for stories that resemble their own, it is also worth considering the use of a clustering technique with the user's own story as an elaborate query.

The possibility of a fine-grained search is something Narrator has to offer that goes beyond the existing websites. In principle all narratives are relevant to all users, because

they are all about patients' experiences with breast cancer. Finding narratives that are more relevant than others will naturally involve rather specific queries.

The question option is the most interesting one. Here it is useful to return passages with a hyperlink to the complete document, so that the user can quickly check whether this document is what she was looking for and on the basis of that decide whether to read it completely.

The question answering in this case is ideally a combination of a categorization and ranking task. Different documents can give different and in some cases opposite answers to a question, because they are about personal experiences. Different things happened to different people, and in addition different people can experience the same thing, e.g. a particular type of medical treatment or daily life situation, in different ways. In many cases it is therefore of benefit to organize the retrieved documents in different categories, for example distinguishing between those authors who had positive experiences with something and those who had negative experiences with it, or, in case of a yes/no-question, documents that answer it affirmatively and those that answer it negatively. In our formal approach, of course, the latter kind of categorization will be easier to achieve than the former. Within each category, ranking the documents for relevance is also useful.

A simpler format to implement is to let the user give a list of search criteria, for example by completing the sentence *I am interested in experiences of people who ...*. On the dots they would then fill in a series of criteria, such as *had radiation, did not have chemo, have a deskjob, do sports, have teenage children, were disappointed in the support they got from their friends*. These criteria can be made into a complex system-internal query straightforwardly, making the most out of our inference technology, while keeping the complexity of the dialogue component to a minimum. It also encourages the user to formulate queries that can sensibly be searched for in individual documents, rather than asking meta questions that would require the system to compare different narratives and perform complex reasoning in order to give an answer. The task is then again a ranking task, with the narrative that is found to fulfill the most criteria ranked highest.

1.2.3 Hybrid retrieval in Narrator

Offering deep semantic parsing of narratives in Narrator is the task of Delilah. Detailed semantic analysis has the advantage of creating possibilities for very precise information retrieval. The success of such precise retrieval depends of course on the quality of the semantic representation and of the retrieval tool that operates on it. Being precise is expected to be crucial in finding, among many narratives about roughly the same subject, those that are most relevant for a particular patient.

Doing inference on deep semantic representations also has disadvantages. It is costly in several ways. The parsing itself can be considered costly, as deep parsing takes up more computer capacity and time than, for example, shallow parsing. The parsing, however can be done off-line and will therefore slow down only the preparation of the system, and not its functioning. Logical reasoning on the basis of complex representations, on the other hand, also needs much computer power. In order to actually retrieve information from these representations, an inference tool is needed that can

operate on them. This happens while the system is running and therefore places a heavy burden on its capacities. In addition, creating the resources required for detailed semantic parsing, such as a lexicon, needs a major investment in terms of man-power. Resources are never complete, which leads to lower robustness.

In Narrator the idea arose to make the retrieval by logical inference more feasible by having a shallower but much more robust application narrow down the search space for the inference tool. Statistics-based approaches to retrieval, in which less or no linguistic knowledge is needed, have the advantage that they are fast and robust. A similar kind of hybrid approach is common in question answering (Jurafsky and Martin, 2000). The difference is that in question answering only selected passages are parsed (on-line), whereas in Narrator the whole corpus is parsed (off-line). Having a corpus that is already annotated with semantic analysis, also means that this semantic analysis can in principle also be taken into account by the robust preprocessing tool.

Some preliminary experiments were performed in the project with Latent Semantic Analysis. LSA is a clustering technique that at least in its basic form disregards all structure and treats each document (that is inputted as a unit) as a bag of words. It compares documents on the basis of the words that occur in them. Words can be weighted differently. Function words are given less weight and important content words can be chosen to be given more weight. Documents are represented as vectors in a multi-dimensional space, where the words are the dimensions. Dimension reduction is applied to reduce noise. Closeness of their vectors (in terms of the angle they make with each other) is then interpreted as similarity of documents. The goal is to form clusters of documents that are similar. Retrieval of relevant text with LSA is done by clustering the query with the other documents.

The idea was that the robust LSA, the performance of which actually improves with an increase in the number of documents it has to cluster, would be able to make a first selection of possibly relevant documents. This narrows down the search space for the logical inference tool, which has problems handling large amounts of data. The inference tool then works on only a limited set of (fragments of) narratives that have been selected by the more robust method. For this pre-selection either the bare text can be taken as input, or the already generated semantic representations themselves, possibly in some derived or adapted form. It is an empirical matter what works best. The LSA step should give a good recall, because documents that do not make it through this first round will not be considered any further. The second step is more focused on precision, perhaps with an option of asking for more results.

Unfortunately the results of the first experiments with LSA clustering of narrative fragments were not very satisfying (Wolf et al., 2006). The experiment did not involve a query yet, but was done to investigate to what extent LSA produced a clustering that made intuitive sense, and if so, whether it was more sensitive to the themes discussed, or to the writing style of the author, or the moment in the disease that the narrative was about. The text fragments were manually categorized according to their most important topics. Author information was already available as meta-data. The LSA clustering did not correspond to this categorization. It did not group fragments together that had a large overlap in the concepts assigned to them by the annotator. Neither did the

clustering turn out to be sensitive to author or moment in the disease. It may be possible to solve the problem by adding more documents or by making different choices in the clustering method. The fact that LSA did provide nice results in other domains, but so far had problems here, also suggests that retrieval on this type of narratives is particularly difficult, and that a bag-of-words based approach is just not sensitive enough. If it turns out that LSA is not able to produce useful clusters with our type of narratives, there probably are other techniques that can do the preprocessing step. This remains for future research.

Whether a hybrid setup of this type in the end is necessary and beneficial depends on the amount of text included in the system and the efficiency reached with an entailment algorithm that works on the new flat logical forms (FLF) that we discuss in chapter four. Inference on FLF will be considerably more efficient than theorem proving on first-order logic. Therefore a hybrid approach may not be necessary for retrieval on a relatively modest amount of text as is the case in Narrator.

1.2.4 The issue of evaluation

The evaluation of information retrieval systems is notoriously difficult and it is one of the main issues addressed in TREC. The Narrator system can be evaluated in terms of user satisfaction, with other ways in which they can learn about other patients experiences as a basis for comparison. This would tell us something about the added value of the system as a whole but not necessarily much about the quality of the retrieval technique we chose as compared to alternatives.

It is in principle possible to let users compare different search options or evaluate different versions of the system. The search engine Hakia, for example, offers users the possibility to compare their results to the results they get with Google. The two ranked lists are then displayed next to each other. A complication is that users in general will be trained in Google-style keyword search, because that is what they have experience with. They will have developed a sense of what makes a good query for this type of search. Most users will however be much less familiar with other ways of searching, such as the alternative we intend to offer in Narrator, which needs a different kind of querying strategy if one wants to make the most of it.

What we do know of logic and linguistics based approaches to inference, and retrieval systems based on these is that they typically have a high precision, but a low recall (e.g. Bos et al. (2007)). This is because a system of explicit logical rules only recognizes an entailment if all relevant information for all reasoning steps is available. Therefore many entailments will be missed, because of some piece of missing background knowledge.

For the kind of semantic annotation produced by Delilah there are no standards for evaluation or comparison. There are no corpora annotated with propositional semantics. Syntactic tree banks are the most elaborate form of linguistically annotated corpora available. On the semantic level, SemEval by SIGSEM recently announced that they will start thinking about a challenge that goes beyond word-disambiguation. The STEP 2008 conference has organized a shared task for comparing semantic representations and exploring the possibility of a theory-neutral gold-standard for semantic annotation (Bos, 2008).

Moreover, there are no competing systems for Dutch that offer a comparable form of semantic analysis.

1.3 Logical textual entailment

The retrieval of text in Narrator is approached as an entailment task. Entailment is the central relation in formal semantics of natural language. Here is a slightly modified definition from Chierchia and McConnell-Ginet (2000), referring to logical form and models containing worlds w, times i and contexts c:

(1) logical entailment
A sentence S relative to Logical Form α entails a sentence S′ relative to Logical Form β iff for every model M = <W,I,<,C,U,V>, c in C, w in W and i in I, if S is true in <M,w,i> relative to α and c, then S′ is true in <M,w,i> relative to β and c.

The organizers of the RTE Challenges offer a practical definition.

(2) generalized entailment
(text) t entails (hypothesis) h if, typically, a human reading t would infer that h is most likely true.

Given an appropriate semantic representation and an appropriate logic, logical entailment must imply generalized entailment. Logical entailment is defined in an operational manner, which is what we need for formal grammar-based retrieval.

In this section, I investigate what kind of entailments should be covered in Delilah/Narrator. In the process of this investigation, I discuss what we can learn from the RTE challenge, and its dataset. At the end of this section I briefly look ahead to the Flat Logical Form that will be discussed in chapter four, and what it means for entailment.

1.3.1 Entailment and implicatures

Zaenen et al. (2005) prefer to call generalized entailment ‘inference’. They distinguish three different classes of inference. One is entailment, which corresponds to logical entailment and is based on lexical and ontological knowledge together with monotonicity. Entailments follow from what is asserted by the utterance.

The second class of inferences are conventional implicatures or presuppositions. Conventional implicatures, as well as entailments, are claims that the author of a text is committed to, although they are not part of what is asserted. The truthconditional status of sentences of which the presuppositions are not met is problematic. In textual inference, however, veridicity plays a much more important role than truth does. It is not possible to decide about the truth of an assertion on purely linguistic grounds. Textual inference systems typically have no means to check statements against the actual world. They only have access to what authors claim, and possibly to the trustworthiness of these authors. Conventional implicatures (presuppositions) are typically computable

provided suitable representations of the words and constructions that introduce them (e.g. appositives, factive verbs, epithets). Blackburn and Bos (forthcoming) show how various kinds of presuppositions can be computed and accommodated in a DRT framework. A complication for their computability is that conventional implicatures cannot be considered to be author commitments when the content they convey has been introduced in a conditional environment (Karttunen and Zaenen, 2005). The example in (3a) does not commit the author to the view that Lance is a Frenchman, whereas (3b) does. If Lance being a Frenchman does not follow from a condition, then it must follow from something the author considers to be a fact.

(3) a. If Lance takes up French citizenship, as a Frenchman he will win the Tour easily.

b. If Lance decides to participate, as a Frenchman he will win the Tour easily.

This means that deciding on the veridicity of presupposed material involves itself an inference task.

Conversational implicatures constitute a third class of inferences. These are can be canceled in context. To the extent that they are computable, it seems reasonable to treat them as inferences provided they are not contradicted elsewhere in the text. Some conversational implicatures are systematic and have a clear source, for example the ones that come with numerals, and let *three*, for example, be interpreted as meaning *exactly three*, rather than as *at least three*. Many other conversational implicatures go beyond what can reliably be computed, as they can arise from what is said and what is not said in a particular context. Especially recognizing the omission of something relevant requires subtle expectations about types of situations and the ways particular people react to them. Even human readers are often unsure about what an author intended to conversationally imply.

Thus, a conventional implicature is something the author says. It is explicit, relevant background information, which the author considers to be true, and which the reader may or may not know already. Therefore conventional implicatures can reliably be used for inference, in addition to entailments, provided a check against information embedded in conditional contexts. They are often a useful source of information in retrieval tasks, and should be part of what an inference engine can handle. A conversational implicature, on the other hand, is something the author does not say, but is likely to mean. Conversational implicatures are less reliable and can be very hard to recognize. Some of the easier ones, introduced by particular words or constructions, can perhaps be included as likely inferences. More than that cannot be expected of formal inference engines any time in the near future.

1.3.2 The RTE challenges

The PASCAL Recognizing Textual Entailment(RTE) Challenges can be seen as representing the state of the art with respect to robust operational textual entailment systems. It was initiated as an attempt to promote an abstract generic task that captures major semantic inference needs across applications, while separating inference from other

problems that different NLP tasks need to handle. The organizers hope to stimulate the development of entailment recognition engines which may provide useful generic modules across applications. So far there have been three RTE challenges (Bar-Haim et al., 2006; Dagan et al., 2006; Giampiccolo et al., 2007).

The data set for the RTE challenge consists of pairs of text and hypothesis. The text is usually a sentence, but up to a paragraph in a subset of the pairs in the third challenge, as it was felt it might be useful to have a little bit more context and stimulate the use of anaphora resolution. The hypothesis is usually a shorter sentence. The task is, given a text and a hypothesis, to determine whether the text entails the hypothesis or not. Common human understanding of language as well as common background knowledge is assumed. The same referring expressions are assumed to have the same reference in text and hypothesis. Tense is ignored. This is a limitation, because in principle one would want an entailment recognition engine to also be able to handle temporal reasoning. Probably this choice was made, because typically more context is needed to evaluate temporal aspects. Ideally one should have access to the whole document that a sentence is embedded in and also know when it was written. This would require a drastic change in the setup of the task.

Even with the present limitations, composing a representative and balanced test set is far from trivial. The pairs for the training set and data set are selected and annotated by humans and designed to correspond to success and failure cases of the actual applications. They are classified by application: information retrieval, information extraction, question answering and multi-document summarization. The collected examples represent a range of different levels of entailment reasoning, based on lexical, syntactic, logical and world knowledge, at different levels of difficulty. The policy was to limit the proportion of difficult cases, but try avoiding high correlation between entailment and simple word overlap between text and hypothesis. From the second edition onwards, text - hypothesis pairs were mostly based on outputs of actual web-based systems on existing application specific benchmarks. This way the challenge could give some sense of how existing systems could benefit from an entailment engine post-processing their output. On the other hand, this of course limits the cases to good and bad results that the applications *have* found. If, for example, these work on the basis of word overlap (e.g. bag-of-words approaches), there will be a bias towards pairs with considerable word overlap. Good results with less or no word overlap that could have been found by a more sophisticated application are then not represented.

The best results got considerably higher with every new edition of the challenge. The baseline of using only word overlap lies around 60%. In the first round hardly any system performed better than that, but in the second edition the results showed for the first time that systems that rely on deep analysis such as syntactic matching and logical inference can considerably outperform lexical systems. Nevertheless, not all deep analysis systems performed above baseline. The main problems seem to be the size of training corpus and a lack of linguistic background knowledge. In the third edition of the challenge the conclusion was reached that machine learning using lexical-syntactic features and transformation-based approaches on dependency representations are well consolidated techniques to address textual entailment. Knowledge acquisition is identified as an

important issue to focus on, to improve systems' performance in future challenges.

The best performing system in the third challenge uses a technique that enumerates a set of propositions (commitments) which are inferable – whether asserted, presupposed, conventionally implicated, or conversationally implicated – from a text-hypothesis pair (Hickl and Bensley, 2007). Preprocessing involves syntactic parsing, semantic dependency parsing, annotation of named entities, pronominal and nominal coreference resolution, and normalization of temporal and spacial expressions to fully-resolved instances. Commitments, taking the form of short sentences, are extracted by means of a series of heuristics. The heuristics fall into five classes: sentence segmentation, syntactic decomposition, extraction of supplemental expressions (such as appositives), relation extraction, and coreference resolution. A word alignment technique selects the most likely matches for each commitment for the hypothesis. Entailment between commitments is then evaluated by a decision tree classifier. Finally an entailment validation module checks whether a hypothesis commitment that was judged to be entailed by a text commitment is in contradiction with any of the other commitments of the text.

Another relatively successful system is COGEX (Tatu and Moldovan, 2007) which uses semantic axioms and a logic prover. Bobrow et al. (2007) present a system with high precision but low recall. One set of rewrite rules generates abstract knowledge representations for text and hypothesis (after LFG parsing and semantic processing), and another set operates on these for entailment and contradiction detection. Their sophisticated representations allow them to distinguish between top-level commitments and embedded commitments. Chambers et al. (2007) introduce a natural logic in which proofs are expressed as incremental edits to natural language expressions. Their system is able to compute effective monotonicity.

1.3.3 The RTE data

The RTE data sets can be a valuable source for how to improve entailment engines, exactly because of the naturally occurring examples. Also, most entailments are quite uncontroversial, as pairs that the annotators didn't agree on with each other were excluded. The data give a nice impression of what problems need to be solved. Vanderwende et al. (2005) analyzed the data set for the first RTE-challenge and found that 37% of the pairs could be resolved by syntax alone, assuming an idealized parser, and 49% could be resolved by syntax plus a general-purpose thesaurus. The question is what is needed for the rest. Vanderwende et al. (2005) only discuss a few non-syntax examples. Some of the examples suggest that more detailed semantics together with a thesaurus that provides FrameNet-like information may provide systematic ways to extend the coverage of the ideal parser-based system. A complex example is (4).

(4) (pair id="287", value="TRUE", task="IR")
<T> The G8 summit, held June 8-10, brought together leaders of the world's major industrial democracies, including Canada, France, Germany, Italy, Japan, Russia, United Kingdom, European Union and United States.
<H>Canada, France, Germany, Italy, Japan, Russia, United Kingdom and

European Union participated in the G8 summit.

One piece of semantic knowledge needed is that a *bringing together* results in a *coming together*. A summit is basically a kind of coming together (of representatives). To participate in the kind of meeting called a summit, is to be (or be represented by) one of the meeting participants. This is all word knowledge, rather than world knowledge. Such examples can give hints on how semantics and thesauri and the way they work together can be upgraded. Another example classified as non-syntax is (5).

(5) (pair id="294", value="TRUE", task="IR")
<T> The three-day G8 summit will take place in Scotland.
<H> The G8 summit will last three days.

Here, I think, simply with a clever semantics, *three-day* can be represented as meaning *lasting three days*, as it is productively used in that meaning. In other cases, it is reported, the annotators considered that there were too many alternations and thesaurus replacements necessary to confidently say that syntax could be used. Another interesting outcome of Vanderwende's (2005) analysis is that many pairs rely on the correct understanding of appositives. These make up 24% of the RTE1 test set. This maybe partly due to the genre used, news-wire like texts, but anyway they seem to have the tendency to be a useful source of information. This again stresses the need to include presuppositions as computed inferences.

In the RTE3 data set I found interesting pairs that depend on disambiguation. For (6) (the next pair in the set is actually very similar) to be solved correctly, *his first film* has to be interpreted as *the first film that he directed/made*.

(6) (pair id="1" entailment="YES" task="IE" length="short")
<T>Claude Chabrol (born June 24, 1930) is a French movie director and has become well-known in the 40 years since his first film, Le Beau Serge, for his chilling tales of murder, including Le Boucher .
<H>Le Beau Serge was directed by Chabrol.

For a human reader this interpretation is hardly avoidable, but it is difficult to pinpoint exactly why. Compare this for example to (7), where *he* is also a movie director, but no one would conclude that this director directed "Lawrence of Arabia".

(7) Since he saw his first film, "Lawrence of Arabia," at a Kabul theater in 1967, when he was 5, his obsession was to make movies.

There has been some controversy about the nature of the data sets. Zaenen et al. (2005) criticize some of the text-hypothesis pairs used. I think that at least for the following pairs they have a point.

(8) T: Hippos do come into conflict with people quite often.
H: Hippopotamus attacks human.
TRUE

(9) T: The White House failed to act on the domestic threat from al Qaida prior to September 11, 2001.

H: White House ignored the threat of attack.
TRUE

A thesaurus stating that *come into conflict with* entails *attack* and that *fail to act on* entails *ignore*, would most definitely cause errors in other cases. It does not even seem very plausible that all conflicts between hippo and human consist of the hippo attacking the human. It is very well possible that the text is intended and in context understood as meaning what is said in the hypothesis. But without context I don't think the entailment can be claimed to hold. It is unrealistic to expect computers to be able to imagine a context in the present stage of research. For the other example, Zaenen et al. (2005) point out that text and hypothesis are naturally understood as referring to the same event of the White House not acting, which is entailed by both. A distinction should be made between entailment and paraphrase. In both cases above, the hypothesis is likely to be an appropriate paraphrase of (part of) the text, but it is not an entailment. Paraphrase is more difficult to compute than entailment, because it is somewhat more subjective and depends on knowledge of typical situations, which cannot easily be attributed to single words or constructions.

Another point that Zaenen et al. (2005) have problems with is that the difference between (10a) and (10b) is not taken seriously.

(10) a. As the press reported, Ames was a successful spy.

b. According to the press, Ames was a successful spy.

(10a) conventionally implicates that Ames was a successful spy, but (10b) does not. This boils down to the difference between statements that hold at text level and embedded statements. Manning (2006) argues that it is reasonable that embedded statements from trustworthy sources are considered veridical, when there is no evidence to the contrary (e.g the author adding that he does not actually believe what is claimed by the source he cites). This means loosening the notion of entailment somewhat, by allowing certain cases of embedding to be ignored. In order to be able to judge when an embedded claim can be lifted to text level and when not, however, it is important to make the difference in the first place and keep track of the sources of claims.

Manning (2006) also points out that modals, such as *can* and *may*, are often hedges and should not block entailment. He takes this to be an argument that the logic applied by semanticists is not always suitable for the task. The difficulty, I think, is to determine when modals are hedges and when they do need to be taken seriously. There certainly are modals that cannot be ignored. There you need the logic, which I think is still a good starting point. This is another case where being too strict about entailments from embedded contexts may negatively affect the results.

Crouch et al. (2006) suggest that the RTE datasets could be improved by more fine-grained annotation along the lines of the KBEval dataset for question answering (Crouch et al., 2005).

1.3.4 The approach for Narrator

Automated inference on logical representations of natural language expressions has thus been experimented with, both in the RTE challenge and in other settings. It turns out that a combination of a theorem prover and a model builder works quite well (Blackburn et al., 2001). Still it is computationally very expensive. When the search space increases, very soon a solution can no longer be found in a reasonable amount of time. It is not difficult to imagine why. Computing entailments in first order logic is a difficult task. Anyone who took a course in first order logic that involved constructing proofs by means of, for example, natural deduction or Fitch-style proofs, will know that coming up with a proof is in general not trivial and requires creative thinking. Typically an exercise consists of one to three premises and a conclusion that is to be reached. Several computer programs have been developed that have the strategies to construct such proofs automatically. But now suppose you have a complete text or even a large corpus of texts represented in first order logic and you want to check whether some hypothesis you have is entailed by any part of the text or by any combination of parts of the text. You would have to check for each proposition and for each combination of n propositions whether they entail the hypothesis. That is, you have to try all possible combinations of premises. And if you want to use additional sources, such as WordNet or the encyclopedia, the set of possible premises gets even bigger. It is clear that inference-based retrieval is a much bigger problem than solving a self-contained logic exercise. In the RTE setting this problem did not come up, because the text was always limited to at most a few sentences.

In Narrator a dual strategy has been developed to remedy this problem. As a preparatory step, the search space is first narrowed down by more robust, shallower techniques. In addition, the actual inferencing is made simpler, through the use of a novel flat notation for the logical form. This Flat Logical Form (FLF) will allow us to use the definition of entailment given in (11). FLF is the topic of chapter four of this thesis.

(11) *generalized entailment*
A text T, represented in FLF entails hypothesis H represented in FLF *iff* every part of H is entailed by some part of T.

Under the assumption that we have appropriate representations, our version of generalized entailment is intended to include strict entailment, conventional implicature and computable cases of conversational implicature. It does not include conversational implicatures that are not reliably computable, nor does it include cases of paraphrase that are not instances of one of the other cases.

1.4 Conclusions

Narrator will offer illness stories on the web with elaborate search facilities. The application is expected to benefit from the high precision that linguistically principled retrieval has to offer. To make optimal use of this, user queries should contain more structure than a list of keywords does. Progress is being made in in the fields of automated

inferencing and of question answering systems based on it. The Delilah parser used in Narrator gives a level of analysis called flat logical form that is expected to facilitate inferencing. Types of entailment that can feasibly be computed are strict entailment and conventional implicature.

The Narrator system has not yet been built since many aspects of it need further research. I will mention a few main points here.

- As yet Delilah's analysis does not go above the sentence level. For example, anaphora resolution across sentence boundaries is at present not covered. Delilah's representations are, however, rich enough to accommodate an existing algorithm, for example a method based on Lappin and Leass (1994).

- The use of other information, such as ontologies is vital for entailment, as the RTE challenge also shows. One might even think of parsing an encyclopedia. Problems in this field are how to trigger the search for external information, and how to use different types of information in different formats.

- The use of meta-data about author and user may be beneficial in the retrieval process. It is at present not clear how best to use such information for the inference based retrieval. Information can be added as propositions to a text, but should they be given a special status?

- For a real working system a dialogue manager is needed[2]. Especially the processing of free text input would require an advanced dialogue manager. This dialogue manager must produce the actual query for the inference engine on the basis of the user input.

- It needs to be investigated if it is useful to have a preprocessing step, in which a fast and robust algorithm selects a set of narratives or fragments that are potentially relevant to the query, such that the inference tool does not need to do all the work alone.

- The inference algorithm itself is yet to be developed.

This chapter served mainly as background. In the rest of the thesis we will concentrate on the semantic representations in Delilah. These representations are designed to be used for automated reasoning, which is one of the techniques that retrieval can be based on. It has been argued here that for certain types of retrieval, including the type needed in Narrator it a is promising technique.

[2]Nap (2008) researched as part of the project how an interface can be optimized for elderly people. As his research assumes traditional search by key-words, it is not directly applicable to a system that offers search possibilities based on deep natural language analysis.

Chapter 2

Delilah: a semantic parser/generator for Dutch

This chapter introduces Delilah[1]. It offers a basic description and discussion and serves as a background for the discussions in the following chapters. A running version of the Delilah system can be consulted at www.delilah.eu.

Delilah is a parser and generator for Dutch sentences. It gives syntactic and semantic analysis for sentences and phrases. Delilah was developed by Crit Cremers and Maarten Hijzelendoorn, and is written in Prolog. A demo of the system is available at www.delilah.eu. The ambition in the Delilah project is to build an operational semantic model of Dutch. The hypothesis is that there is a compositional basis of meaning which is computable, a research programme that goes back at least to Montague (1973).

In Narrator, Delilah is used to parse the narratives and provide them with formal semantic representations. The semantic representation of narratives is done by the parser, but since the same grammar is used for both parsing and generation, it is important to also keep in mind possible consequences for the generator, when making a change to improve parsing. Still, I will focus mostly on the parser.

First, I briefly introduce the grammar that Delilah uses. This is a description of the existing system with some reflection. Then, I explain the structure of the lexicon. This section too consists of a discussion of the existing system. The largest part of the chapter is devoted to the semantics. I explain the basic principles and discuss a variety of problematic issues. I will point out what Delilah can already do, what the problems are and in which ways things could be improved. The first two subsections are mainly description. In subsection 2.3.3 I explain how different scope readings are obtained now and how that could be improved. In subsection 2.3.4 I discuss whether scopal operators other than generalized quantifiers should be stored. Now this is not the case. I ague that this kind of storing is desirable only for the negation in negative indefinites that give rise to split scope readings. Subsection 2.3.5 is a more general discussion about scope disambiguation and underspecification. In subsection 2.3.6, I discuss a problem in the existing treatment of adjuncts and propose an alternative. Subsection 2.3.7 and 2.3.8 and 2.3.12 mention some limitations of the system in terms of coordination, anaphora

[1]Delilah is not an acronym, but the successor of SAMSON (SAMenStelling van ONgelijk gerichte functoren).

and disambiguation. Subsection 2.3.9 explains the approach to extended lexical units. Subsection 2.3.11 briefly introduces the main characteristics of our event semantics, which is discussed elaborately in chapter three. In subsection 2.3.10, I criticize the way concepts are represented in the present system. Section 2.4 deals with robustness issues. In the final section, I mention some other computational semantics systems.

2.1 The grammar

This section explains the grammar as far as it is necessary for a good understanding of the way the lexicon and the semantics are built up. The grammar rules guide the unification of complex graphs provided by the lexicon.

Delilah's grammar is a variant of Combinatory Categorial Grammar (CCG), based on Steedman's Generalized Composition (Steedman, 1996). The backgrounds of these formalisms are discussed by Cremers (1999b; 2004). The grammar was argued to be mildly context sensitive by Cremers (1999a). van de Woestijne (1999) developed a chartparser to apply the grammar more efficiently than the original parser did. The particular version of Categorial Grammar used in Delilah is not the only grammar formalism that is suitable to support compositional semantics. There are many others, see the overview of other systems in section 2.5. The formalism used was developed to resolve the 'disharmonious' Dutch verbal cluster without recourse to full context sensitivity.

Each word has a syntactic type that tells how it combines with other words and phrases. That is the basic principle of Categorial Grammar. Syntactic types in Delilah are rigid. Ambiguity between different types is handled in the lexicon. When a word has n different types, it has (at least) n different entries in the lexicon. The grammatical rules are unambiguous. The grammar does not provide for type shifts in the derivation. That is, there are no syntactic rules that change one type into another one. This also means that every rule is a rule of composition, taking exactly two types as input and giving exactly one type as output. Each complex type has the following form[2]:

(12) stack of left arguments($l_n \ldots l_1$) \head/ stack of right arguments ($r_1 \ldots r_m$)

The head is a primitive category, such as `np`, `vp` or `s`. The arguments in the lists are addressed by their heads, but represent complex types of the same form, i.e. a head with two (possibly empty) argument lists. The argument lists can be seen as an agenda. Of each stack, the first argument has to be canceled before the next argument becomes available for canceling. The left arguments have to be found to the left of the head and the right arguments have to be found to the right. The first argument in a list is to be found first, i.e. closest to the head. Two categories combine if the secondary category can

[2] In the actual implementation the notation is different, namely:

(1) head\stack of left arguments/stack of right arguments

The adapted notation used in this book, benefits human readability, because the order of the arguments directly corresponds to actual word order.

be canceled against the relevant argument of the primary category, i.e. in the following configurations[3]:

(13) a. PLA\prim/[sec|RestPRA] SLA\sec/SRA
b. SLA\sec/SRA [RestPLA|sec]\prim/PRA

In the first case an element with head category "sec" is on top of the right argument stack and a phrase headed by this category is indeed found on its right. The second case is the same, except that the argument is sought and found on the left. In each case the composition has four possible outcomes, depending on how the remaining argument stacks of the primary and secondary category are combined. The result of composition is a new complex type, where the head is still the primary category, the argument consumed is canceled from the agenda and a new agenda is composed from the arguments that were still on the agenda of the secondary category and the remaining arguments on the agenda of the primary category. The new agenda is formed by appending argument lists. The lists are never mixed, neither is the internal order of the lists altered. The two left lists are appended to form a new left list and the two right lists are appended to form a new right list. Arguments cannot be moved from the left list to the right list, or the other way around. In this way directionality is preserved. The different possibilities are demonstrated below. This mechanism is called Extended Generalized Composition.

(14) PLA\prim/[sec|RestPRA] SLA\sec/SRA ⇒ SLA+PLA\prim/RestPRA+SRA
PLA\prim/[sec|RestPRA] SLA\sec/SRA ⇒ PLA+SLA\prim/RestPRA+SRA
PLA\prim/[sec|RestPRA] SLA\sec/SRA ⇒ SLA+PLA\prim/SRA+RestPRA
PLA\prim/[sec|RestPRA] SLA\sec/SRA ⇒ PLA+SLA\prim/SRA+RestPRA

SLA\sec/SRA [RestPLA|sec]\prim/PRA ⇒ SLA+RestPLA\prim/PRA+SRA
SLA\sec/SRA [RestPLA|sec]\prim/PRA ⇒ RestPLA+SLA\prim/PRA+SRA
SLA\sec/SRA [RestPLA|sec]\prim/PRA ⇒ SLA+RestPLA\prim/SRA+PRA
SLA\sec/SRA [RestPLA|sec]\prim/PRA ⇒ RestPLA+SLA\prim/SRA+PRA

In practice, however, out of the possible compositions in (14) only the two repeated in (15) are used; one for canceling an argument on the right and one for canceling an argument on the left.

(15) PLA\prim/[sec|RestPRA] SLA\sec/SRA ⇒ PLA+SLA\prim/SRA+RestPRA

SLA\sec/SRA [RestPLA|sec]\prim/PRA ⇒ SLA+RestPLA\prim/PRA+SRA

These rules parse the following configurations:

(16) a. PrimArgsL **SecArgsL** prim sec SecArgsR PrimArgsR
b. **SecArgsL** PrimArgsL sec prim PrimArgsR **SecArgsR**

[3]PLA = Primary Left Arguments, PRA = Primary Right Arguments, SLA = Secondary Right Arguments, SRA = Secondary Right Arguments

The bold faced arguments are separated from their head. The discontinuity effect disappears if these argument lists are empty at the time of composition, i.e. if the secondary category has already consumed its arguments, or never required any, on the relevant side[4]. Note that the primary head cannot be separated from its arguments in any interesting way by the secondary head, because the secondary head is one of its arguments. One of the cases in which arguments are separated from their head are the Dutch verbal clusters with their crossing dependencies. Another such case is the more common *wh*-fronting.

An example sentence for which this kind of discontinuity is needed is (17)[5]. It is easy to see that, unlike in the English translation, none of the np arguments is adjacent to its head; *De nijlpaarden* is the argument of *voeren*, *Alice* is an argument of *helpen*, *Bob* is an argument of *zien*, and *ik* is an argument of *heb*.

(17) ...dat ik Bob Alice de nijlpaarden heb zien helpen voeren.
...that I Bob Alice the hippos have seen help feed
'that I have seen Bob help Alice feed the hippos.'

Below, the same sentence is given, but now with the types for each word instead of glosses. Heads are in boldface. The backslash '\' introduces left arguments to the left of the head, the forward slash '/' introduces right arguments to the right of the head, and empty argument lists are left out. I have numbered the nps so that it is easier to see what they will turn out to be arguments of. The unfamiliar type s_vn is the head category of a finite verb in a subordinate clause. This is different from the finite verb in a main clause because of the different word order. The category of a subordinate clause is abbreviated as s_sub. The nps would be called DPs in modern versions of generative grammar.

(18) ...dat ik Bob Alice de nijlpaarden heb zien
s_sub/s_vn **np**$_1$ **np**$_2$ **np**$_3$ **np**$_4$/n **n** np$_1$**s_vn**/vp np$_2$**vp**/vp
helpen voeren.
np$_3$**vp**/vp np$_4$**vp**

To get the right parse, the verb cluster has to be formed first. Combining *helpen* with *voeren* results in a type np$_3$,np$_4$**vp** for the string *helpen voeren*. The vp argument of *helpen* is canceled against the vp head, introduced by *voeren*. The newly formed type keeps the vp head from *helpen* and its agenda is composed of the unfulfilled agenda items of both composing types. Both were still looking for an np to the left. The np argument of *voeren* is put first on the new agenda, followed by the np argument of *helpen*. This is in accordance with the first rule in (15). In a similar way *zien helpen voeren* gets the type np$_2$,np$_3$,np$_4$**vp** and *heb zien helpen voeren* gets the type np$_1$,np$_2$,np$_3$,np$_4$**s_vn**, the full cluster. Then, all of the np arguments can be consumed in the right order. Of course *de* and *nijlpaarden* first have to form an np together. This np then, np$_4$ is the first to be found and luckily it is also the one on top of the list. So, the string *de nijlpaarden heb zien helpen voeren* is of the type

[4]The following type of discontinuous configuration is not supported: PrimArgsL **SecArgsL** prim sec PrimArgsR SecArgsR.

[5]A subordinate clause was chosen, to avoid the V2 effect and get a full-blown verbal cluster.

$\texttt{np}_1,\texttt{np}_2,\texttt{np}_3\backslash\textbf{s\_vn}$. In the same way the other np arguments are consumed, resulting finally in a type s_vn that does not need any more arguments. This s_vn is then consumed by the type of *dat* to form an s_sub, a subordinate clause. The full derivation is represented schematically in figure 2.1.

The rules of composition in their most general form cover the wildest form of verbal clustering, while leaving it optional. The rules in (15) would also allow for the English word order, where all the verbs combine with their np arguments, before combining with each other. At the same time the rules also wouldn't object to a word order in which *nijlpaarden* is separated from its head *de* ending up to the right of the verbs. This would happen if *de* could be consumed by the verb cluster, before having consumed its own argument. Such variants of (17), however, do not occur in Dutch. Therefore, more restricted instantiations of these rules are needed for the specific grammatical configurations that occur in Dutch, to ensure, for example, that verbal clustering is obligatory and that nouns stay next to their heads.

Restrictions are imposed on composition through modes of composition. In a lexical entry, for each argument the mode is specified, under which this argument can be consumed. Determiners will take their n arguments under a different mode, than clustering verbs take their vp argument. The mode adds additional requirements to (one of) the rules in (15). A common requirement is that particular stacks be empty, which limits the possibilities for discontinuity. Also, a list may be required to only (or at most) contain an argument that is to be canceled under the mode 'wh'. This means that any other argument on the list must have been canceled before this composition and that only a wh-argument can be on the list that will be separated from its head. In principle an argument list could also be required to be non-empty, but this doesn't happen as such.

What does occur is the requirement that an argument, or all arguments have to have been canceled before the present composition. This is marked by flags. Every argument list has a flag. There are three different flags; 0, 1 and wh. The 0 flag indicates that no argument from the list has been canceled yet. Flag 1 indicates that one or more arguments from the list have already been canceled. So, requiring an empty list with flag 0, means that there must never have been any arguments in that list, whereas requiring an empty list with flag 1 means that there must have been at least one argument. The 0 flag is used, for example, in verb clustering, where all arguments of the verbs have to occur outside the verbal cluster. The wh flag indicates that a wh-argument has been canceled, which means that the 'SpecCP' position is filled. This is relevant for the Dutch V2 effects. The position before the finite verb can be filled only once. It is called *wh*, because this is the same position that is the landing site for *wh*-movement (the fronting of question words). When two argument lists are appended into one, this new list normally gets a flag according to what the flags of the composing lists were, but exceptions are possible. Also, when an argument from a list is canceled, the flag of the list minus that argument should be 1. But also here manipulation is possible. For example, when a verb consumes its separable particle on the left, this does not change the flag of its argument list. This way the particle+verb combination, for example *invullen* 'to fill in', can still enter verbal clustering. The use of flags is therefore somewhat opportunistic.

A lot more criteria could in principle be imposed on argument lists. It would be

<page_header>30 Chapter 2. Delilah: a semantic parser/generator for Dutch</page_header>

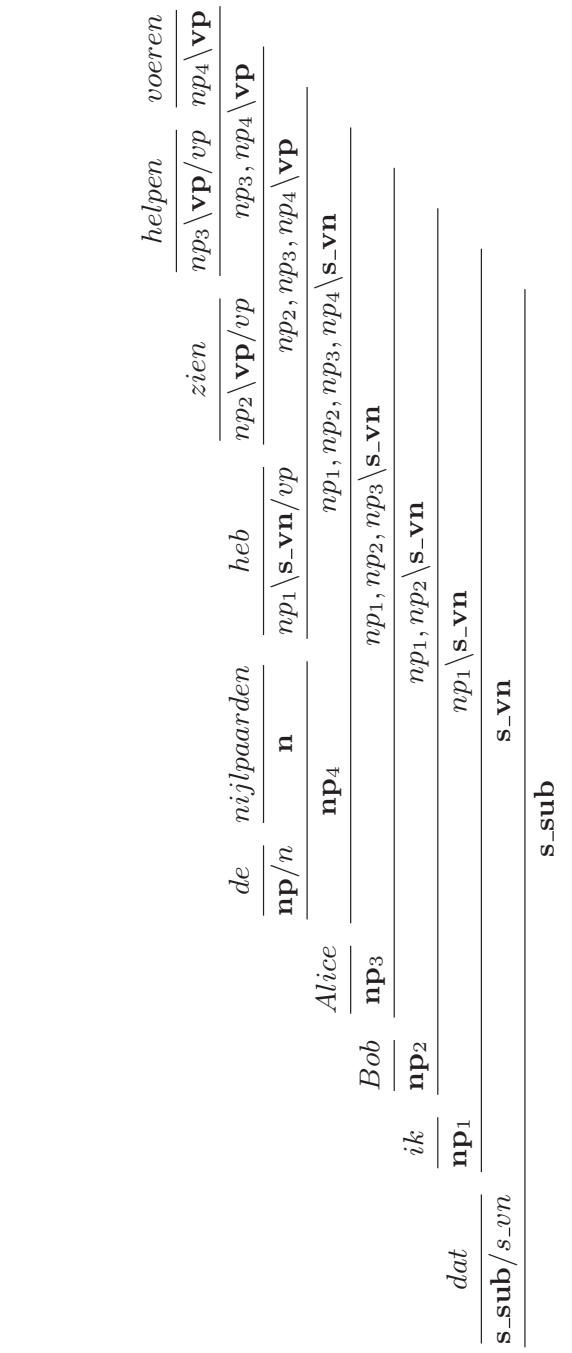


Figure 2.1 — Type-logical derivation of (17)

possible to exactly specify the arguments that should occur in the a list with their modes and their relative order. This, however, would make the rules very un-general and multiply the number of modes needed. Fortunately, it turns out that rules can be fairly general, and that only about a dozen different ones are needed. Actually the only requirement on the content of argument lists that occurs next to the ones already mentioned, is that certain modes do not occur in the list. There is a standard check that can be called for this purpose, since it is always the same set of modes that is disallowed.

Earlier, I said that argument stacks can only be appended, not mixed with each other and that the order of arguments cannot be changed. Stacks that contain a `wh`-argument (these are always left argument stacks) form an exception to this. The `wh`-argument always has to be at the bottom of the stack, also after composition, because it is to be the most leftward argument. Treating `wh`-arguments as a special case in this way is like having a separate list for `wh`-arguments.

There is one other exception that effects the order of arguments. Arguments of the category `rnp`, a special category for the particles *er* and its stressed counterpart *daar*, which mean something like 'there', can be canceled 'before their turn'. An `rnp` argument can be consumed if it occurs anywhere in the argument stack, not only when it is on top. This goes against the idea of stacks and makes the grammar less restrictive. The more arguments can do this, the less the order of the argument list will matter. At some point all permutations will be possible. It would therefore be much better if this rule were not needed and the phenomena could be handled in some other way.[6]

Some rules only apply to types with particular heads. For example in the `rnp` rule mentioned above the secondary head must be of the category `rnp`. Sometimes rules are differentiated for different head categories, for example, if the head is of a particular category, an additional requirement may be imposed, which otherwise does not apply.

Changing the category of the head may be possible in the format of the rules, but is not meaningful. The category of the secondary head plays no role anymore once it has been consumed. In other words, it does not occur in the output type. The head of the output type is the same as the head of the primary input type. If a head of category x in the primary input type were changed into a head of category y in the output type, this would give the same effect as when it had always been of category y and the rule asked for a category y as the head of the input type. The category of the primary head only becomes important if after composition the result will act as an argument for another category, that is if in one of the next composition steps it will play the role of secondary head. Composition was designed to be input-sensitive: exactly one occurrence of a head in the input is canceled against exactly one occurrence of that head in the argument stack of the primary head. It may be possible to consume a category from the input without canceling the corresponding category on the agenda, in such a way that another such category in the input can be consumed. This would relax input-sensitivity. Also, the argument on the list will at some point have to be canceled, otherwise the derivation will not be able to terminate. I discuss this in the section about adjuncts 2.3.6.

What is certainly not desirable is if one or more arguments are added to the agendas in

[6]The placement of `rnps` is notoriously complicated and no solid set of rules has been developed for it so far.

composition. Also arguments that are on the agenda should not be altered, e.g. replaced by others. Such operations could make the grammar Turing-complete.

The types to enter the rules are provided by the lexicon. Normal lexical items introduce exactly one complex type. *Wh*-words, however, introduce two types. For example *wat* 'what' introduces the type `[]\q/s`, something that forms a q(uestion) if it finds something of the category s(entence) on its right, and the type `[]\np/[]`, a noun phrase (or actually a determiner phrase) that no longer needs any arguments. *Wat* combines with something of type `np^wh\s/[]`, a sentence that is still waiting to find an `np` to its left, which should be canceled under mode `wh`. (This `np` may have started out as an argument of a category embedded under `s`.) This type can then consume the `np`, introduced by the question word to form a complete `s`, which then in turn can be consumed by the type headed by `q`, to form a question.

In summary, the grammar rules are based on only one rule with a right and a left variant. Different instantiations of this rule, all with their different restrictions, are invoked by lexical constructions through the modes of composition. The restrictions mainly control the order in which the compositions take place. The most important tools for doing this are flags, emptyness requirements on lists, special treatment of wh-items and a check for the occurrence of modes that have to be executed first.

Delilah's grammar, with its different modes of composition bears similarities to Moortgat's Multi-Modal Categorial Grammar (MMCG) (Moortgat, 1997). The most important difference is that MMCG stays closer to the traditional CG in the sense that type-lifting rules are applied in the grammar and thereby scope ambiguities are already covered in the syntactic derivation. Each reading has a different derivation. In Delilah on the other hand semantic underspecification of scope relations is used.

2.2 The lexicon

The complex graphs of which the grammar guides the unification are provided by the lexicon. Which heads take which arguments under which mode is part of the information contained in the lexicon. Lexical items are complex graphs of feature structures: attribute-value matrices, which are created by lexical rules on the basis of lemmas and templates. Graphs of lexical items unify with each other, until one big graph for the whole sentence is formed, which is then the fully specified parse tree for that sentence. Graphs are unified if the grammar allows for composition of their types and their feature structures allow for unification.

In (19) we see one of the actual entries for *lezen* 'to read' in an infinitival form.

(19)
```
|ID:A+B
|HEAD: |CONCEPT:read
|      |PHON:lezen
|      |LOG:read
|      |SYNSEM: |ETYPE:event
|      |        |FLEX:infin
|      |        |VTYPE:transacc
|PHON:C
```

```
|PHONDATA:lijnop(lezen,A+B,[arg(left(1),0,D)],C)
|LOG:{{[E*(B+99)#F, G*(B+H)#I,
λJ.∃K.read(K) & event(K) & J*(A+B)#L], [], []},
λM. agent_of(L, F) & theme_of(L, I) & attime(L, M)}
|SYNSEM:|CAT:vp
|         |EVENTVAR:L
|         |EXTTH:agent_of~[A+B,F]
|         |PREDTYPE:nonerg
|         |TENSE:untensed
|TYPE:0~[np^0#B+H]\vp/0~[]
|ARG:|ID:B+99
|    |PHON:N
|    |LOG:E
|    |SYNSEM:|OBJ:subject_of(A+B)
|    |       |THETA:agent_of
|ARG:|ID:B+H
|    |PHON:D
|    |LOG:G
|    |SYNSEM:|CASE:obliq
|    |       |CAT:np
|    |       |OBJ:dirobject_of(A+B)
|    |       |THETA:theme_of
```

Since this is an infinitive – sentence final in Dutch – it looks for an object NP on the left. This is indicated under TYPE. This object argument is to be canceled under the conditions of mode 0, indicated by ^0, which refers to a particular mode of composition (see section 2.1). The ID feature is used for building the tree. In *ID:x+y*, *x* refers to the higher, directly dominating node. As one can see, all arguments of the verb start with B, which is the second variable in the value of the ID feature of the VP. The NP argument in the left argument list has index B+H, which is the object. (An argument of this argument, for example, will start with H.) The ID feature also links the semantic items in the store to the syntactic constituents they originate from. I will talk elaborately about the semantics in the next section[7]. For now it is useful to know that the semantic field has the form:

(20) {{[QuantifierStore], [PronounStore], [ReflexiveStore] }, Body}

Only the quantifier store and the body are important at the moment. The semantics of the arguments is stored in the quantifier store. Stored items have the form Sem*(ID1+ID2)#X, where Sem is the semantics of an argument, ID1+ID2 refers to the constituent of origin, and X is the variable this item binds in the body.

The data structure is very similar to those used in Head-Driven Phrase Structure Grammar (HPSG) (Sag and Wasow, 1999), except for the use of variables (capital letter) instead of numbered boxes. Also Delilah's data structures stay close to Prolog, which does not provide the kind of large brackets used in attribute-value matrices. But one could read the dotted lines as indicating which parts of the structure should be in such

[7] Minor adaptations have been made to the semantic notation, for the sake of readability.

brackets. With respect to the features and values that can be used, Delilah is more liberal than HPSG, but it has a rigid grammar underneath.

Entries like the one in (19) are constructed on the basis of lemmas, with the help of templates, by rules. The most basic rules mainly just combine the information in the lemma and the template. But there are also rules that produce derived forms, such as the finite forms of verbs. The lemma contains information that is specific to the particular word, such as concept and phonological form. The template contains information that a class of words have in common. The use of templates is mostly practically motivated. Information that is the same for a whole class of words only has to be entered once. The entry we have seen in (19) was created on the basis of the lemma in (21) and the template in (22).

In (21) we see the lemma of *lezen* 'to read'. The predicate lemma/5 always has the following five arguments: The first argument is the name of the lemma, which for convenience normally corresponds to a form of the word. The lemma name has to be an atom. The second is its broader class, such as *verb*, *noun* or *adjective*, which is relevant for the lexical rules that I discuss later in this section. The third argument is a list of one or more names of templates that the word is based on. The fourth argument is a list of specifications, in the form of paths, to be added to the template in order to 'personalize' it for this particular word. And the fifth and last argument is a (possibly empty) list of irregular derived forms.

(21)
```
lemma(lezen,
      verb,
      [trans_v, trans_v_sc, trans_ssub, trans_qsub],
      [arg(ID+_ID1+1):synsem:theta:theme_of,
      arg(ID+_ID3+10):synsem:theta:agent_of,
      head:synsem:etype:event, head:phon:lezen,
      head:concept:read, head:sem:read],
      [pastsing:las, pastplur:lazen,
       participle:gelezen] ).
```

The verb *lezen* is here assigned four different templates. The first one is the template for transitive verbs, given in (22).

(22)
```
template( trans_v, verb,
[ id:Top+ID,
synsem: [cat:vp, tense:untensed, predtype:nonerg,
         eventvar:EV, extth:Stheta~[Top+ID, A]],
sem: {{ {[SemS*(ID+ID1)#A, SemO*(ID+ID2)#B,}
     λEStructure.∃E.Main(E) & Etype(E) &
     EStructure(Top+ID)#EV],[],[]},
     λTime.  Stheta(EV,A) & Otheta(EV,B) &
     attime(EV, Time)},
head: [phon: _X,
       synsem:[vtype:transacc, flex:infin, etype:Etype],
       sem: Main],
arg(ID+ID1+10): [phon:_Subj,
```

```
                    synsem:[theta:Stheta,
                            obj:subject_of(Top+ID)],
                    sem:SemS],
    arg(ID+ID2+1): [phon:_Obj,
                    synsem:[obj:dirobject_of(Top+ID),
                            theta:Otheta, cat:np,
                            dir:left(1), mode:0, case:obliq],
                    sem:SemO]
    ]  ).
```

The verb takes two arguments, indicated as `arg(ID+ID1+10)` and `arg(ID+ID2+1)`. The numbers `10` and `1` are used to identify the arguments when integrating the information in the template with the information in the lemma, since the actual `id` numbers are variables. The basic position of the argument with respect to the head is prescribed in the form of `dir:left(1)`. (The direction can be left or right and the lower the number between brackets, the closer the argument in question stays to the head. Thus these numbers encode the order of the arguments.) Rules that produce the different forms of the verb may modify this position. In the template, the mode of composition to be used is indicated by `mode:0` in the `synsem` specifications of the argument.

The values left variable under `head` are to contain information contributed by the lemma, namely: the phonological form of the head, the event type (so far we distinguish between events and states) and the semantics of the head, which is used in the semantics of the whole VP. Also the participant roles of the arguments are contributed by the lemma and used in the semantics.

For the infinitive form above, the only thing the rule changed in the graph is the second `id` value of the subject. It is changed from a variable into a constant, because the variable would otherwise have to remain free, which means we'd have to be very careful to not let it be instantiated by accident at some point. The subject will grammatically be an argument of the auxiliary. Through a 'control relation' its semantic value is unified with the external theta role (in this case agent) in the semantics of *lezen*. The combinatory type is just constructed out of the information on categories and directions.

One other variant of the infinitival entry is made, namely one where the object (the only argument on the agenda here) is marked for *wh*. This is to cover sentences like (23), where the object of *lezen* 'read' is *wh*-extracted.

(23) Welk book heeft Alice altijd al willen lezen?
which book has Alice always already wanted read
'Which book has Alice always been wanting to read?'

Quite a bit more happens when the finite forms are being constructed. First of all the subject gets a category, mode and direction, which make it a grammatical argument. Second, a tense-pronoun (see 2.3.11) is added to the quantifier store (see 2.3) and a tense operator is added to the body of the semantics. The head category is changed from `vp` to `s_vn`, the category of a finite verb in a subordinate clause. Also person and number features are added. The phonology (or actually spelling) of the word

forms that correspond to these features is provided by a morphological component. The combinatory type is constructed on the basis of the information in the changed graph (i.e. now the head category is `s_vn` and there is also a subject argument on the agenda). The result are the entries for finite verbs in subordinate (i.e. verb final) clauses. On the basis of these, the entries for verbs in main clauses are constructed, where they are in second position. Here the head category changes to `s` and the direction of all arguments is changed into rightward, because now that the verb is in second position all (non-*wh*) arguments occur to the right of the verb. Also the modes of some arguments are changed, because the verb in second position does not participate in clustering. Of all entries (including the `s_vn` ones) *wh* variants are constructed. One for each argument that can be extracted. Of the third person forms, variants are made with an added existential expletive subject (*er*). Next to the `s` and `s_vn` entries, also entries are formed with the type `q`, for yes/no-questions. Here the modes of all arguments are changed to modes that do not allow any extraction, since these questions are verb-initial, and a question operator is added to the semantics. Also imperatives are formed, with no syntactic subject and the semantic subject set to the constant that represents second person. (25) is an example of one of the entries produced by the finite-making rules. It is the entry for the first person present, with a *wh*-subject, as it occurs for example in (24).

(24) Ik lees de krant.
I read the newspaper
'I read the newspaper.'

(25)
```
|ID:A+B
|HEAD:|CONCEPT:read
|     |PHON:lees
|     |LOG:read
|     |SYNSEM:|ETYPE:event
|     |       |FLEX:fin
|     |       |NUMBER:sing
|     |       |PERSON:or([1,2])
|     |       |TENSEOP:at-pres
|     |       |VTYPE:transacc
|PHON:C
|PHONDATA:lijnop(lees,A+B,[arg(right(-1),0,D),
                   arg(left(11),wh,E)],C)
|LOG:{{[F*(B+G)#H, I*(B+J)#K,
λL.∃M. read(M) & event(M) & L(A+B)#N,
pron(A+B)*(A+B)#O], [], []},
AtPres(O).agent_of(N, H) & theme_of(N, K) & attime(N, O))}
|SYNSEM:|CAT:s
|       |EVENTVAR:N
|       |EXTTH:agent_of~[A+B,H]
|       |PREDTYPE:nonerg
|       |SUBQMODE:P
|       |TENSE:tensed
|TYPE:0~[np^wh#B+G]\s/0~[np^0#B+J]
```

```
|ARG:|ID:B+G
|    |PHON:E
|    |LOG:F
|    |SYNSEM:|CASE:nom
|    |       |CAT:np
|    |       |NUMBER:sing
|    |       |OBJ:subject_of(A+B)
|    |       |PERSON:1
|    |       |QMODE:P
|    |       |THETA:agent_of
|ARG:|ID:B+J
|    |PHON:D
|    |LOG:I
|    |SYNSEM:|CASE:obliq
|    |       |CAT:np
|    |       |OBJ:dirobject_of(A+B)
|    |       |THETA:theme_of
```

When comparing this entry to the infinitival entry, we see that the `phon` value is different, the value of `flex` has changed to from `infin` (infinite) to `fin` (finite), an extra pronoun has been added to the store in the semantics and the tense operator `atpres` has been added to the body. The category has changed to `s` (`SYNSEM:CAT:s`). The subject has gotten a category, namely `np`, and it is required to be 1^{st} person. The type now has a *wh*-subject `np` on its left agenda, whereas the object `np` has been moved to the right agenda.

In a similar fashion as the entries for infinite and finite forms are constructed, participles are also derived by a set of rules. I will not go into details, but as one can imagine in passive participles the object is made the external argument, etc.

It follows that for each verb form in each configuration, a different entry is needed. This creates a very large lexicon, but that is not a problem, as the lexicon is indexed in such a way that it can be searched in linear time. Before Delilah can work, all the lexical entries need to be created and written to a disk, organized according to the paths that they contain. This lexicon is searched during parsing and during generation. The storage and retrieval of lexical items was discussed in Hijzelendoorn and Cremers (2007).

The lexical rules can in principle change anything in the graph they get as input. Arguments can be removed, added, altered; categories can be changed. The semantics can be manipulated, as happens with the introduction of tense- pronouns and operators or question operators. This means that these rules are rather unrestricted. As long as the rules create a finite lexicon, however, this is not a problem. So, a rule that adds an argument, such as expletive *er*, is okay, but a rule that recursively adds arguments, is not, because then the procedure that creates the lexicon will never terminate.

The alternative to having these rules, would be specifying much more by hand, on the lemma and template levels. For example, instead of using a set of rules to create the entries for the finite forms of verbs, one could make different lemmas for the different verb forms and link each of them to a number of templates to cover the different configurations. This would be highly impractical. The advantage of rules is that

they capture generalizations in the lexicon.

As for the division of labor between lemmas, templates and lexical rules, rules are meant to be general. The set of rules to make finite forms applies to all verbs. This does not necessarily mean that all words have to be treated exactly the same. A rule can check for a certain feature and treat words with that feature differently. For example, the rule that makes passive participles only works on verbs in which it finds a specification for the direct object. This functions as a check on whether a verb can be passivized. There is, however, no clear dividing line between what should be arranged by lemmas and templates and what should be captured in rules. Plural nouns, for example, used to have their own lemmas and their own template. Now they are derived from their singulars. This, of course, does make the situation more consistent than it was before. It goes towards letting the rules take care of exactly those cases that can be described as the production of flectional paradigms. These are in general also the forms that are produced by the morphological component.

Templates can be seen as representing some of the more general constructions that individual entries inherit from. However, contrary to the view on constructions in (at least most versions of) Construction Grammar (Croft, 2001), information in the lemma can overrule information in the template, which is useful in the case of exceptions. Also, not all abstract constructions that are assumed to play a role in Construction Grammar are represented by templates. A lexical entry like (25) is usually assumed to inherit not only from the transitive verb construction, but also from the first person construction. Constructions like the first person construction, however, are at best represented by the rules that derive these forms. A maybe more interesting construction grammar view on the lexicon, would be to say that each lexical entry is a construction. The more general constructions can then be extracted on the basis of the paths that lexical entries have in common. For example, (25) shares many paths with other entries for transitive verbs and also with other entries for first person present forms. The paths of all lexical entries can be collected into one huge feature-value graph. This graph can be taken to represent the inheritance network of relations between the different lexical entries.

The decision to make a separate template for a class of words (i.e. identifying it as a distinct class) can depend on several things. A common criterion is the category of the head together with the number of arguments and the categories of these arguments. Thus, we have templates for intransitive verbs, transitive verbs, verbs with a PP complement, verbs with a sentential complement, count nouns without complements, nouns with a sentential complement, etc. The semantics also plays a role. Number adjectives, for instance, have a template that is different from that for normal adjectives, because they have a particular semantics. In a template the semantic structure of the 'construction' is pre-programmed. Only some values still have to be filled in. The only way to change that structure from the lemma, is to overwrite it completely. Specifying the full semantic structure in a lemma is to be avoided as much as possible, because it is labor intensive and mistakes are easily made. So, if a group of words shows the same pattern, it soon becomes worthwhile to dedicate a template to this group.

When an argument does not play a role in the semantics, it is not too much of a problem to encode it as an extra argument in the lemma. Obligatory reflexives are now

encoded that way. The *zich* 'self' in *zich schamen* 'to be ashamed' is an obligatory reflexive pronoun, since the verb always requires the presence of a reflexive pronoun that agrees with the subject. The reflexive argument is added by the lemma to the arguments that are already in the template. The semantics doesn't change. Since the reflexive pronoun is always there, its contribution to the semantics of the combination does not need to be recorded separately. Verbs with separable particles (e.g. *opbellen* 'to phone', where *op* is the particle), on the other hand, do have their own templates, even though the particles do not contribute to the semantics. They are quite frequent, so the alternative would mean that a lot of lemmas would need to have a particle added. It does however double the number of verbal templates, because particles co-occur with any combination of other arguments. It may be an option to let the particles be added by the rules. If a particle is specified (with its phonology) then the rule should construct the actual argument for it. The basic position, directly to the left of the verb and the mode are always the same.

Sometimes there are two versions of the template, for example for transitive verbs with an additional PP complement. Here the direct object and the PP can occur in either order. This is not general enough to arrange in the rules. Therefore there is a version of the template for each order.

In summary, templates are general constructions, rules create the paradigms for these constructions and lemmas are individual words, marked for which constructions they can head. All items in the actual lexicon that is used for parsing and generation are the output of rules. Lemmas and templates are their input.

Also collocations are interesting in the context of lexical organization, but I will postpone discussing them till after I have explained the semantics better.

2.3 The semantics

This section aims at explaining how the semantic representation works. Two different levels of semantic representation can be distinguished in Delilah, Stored Logical Form (SLF) and Logical Form (LF). SLF is a direct result of unification and is obtained during the derivation. It is the value of the attribute `LOG` in the graph. Scope relations are still underspecified in SLF. LF is a formula in first order predicate logic, that is constructed post-derivationally on the basis of SLF and possibly additional information in the template. In LF scope relations are fixed. One SLF can give rise to several LFs. In chapter three a new semantic output format is discussed. To derive this new form, some minor changes needed to be made. LF is then replaced by, on the one hand the new, flatter representation and on the other hand a predicate logical form that is very similar to the original LF. Since the changes are not relevant for the normal LF, I will not discuss them until chapter three.

2.3.1 Stored Logical Form

The simple sentence in (26) will be used to illustrate the most important aspects of semantic composition and application. The notation of semantic structures is somewhat

simplified here for readability. I still use capital letters for variables.

(26) Elke man slaapt.
every man sleeps
'Every man sleeps/ is sleeping.'

This sentence consist of a subject *elke man* and a verbal predicate *slaapt*. The subject *elke man* in turn exists of the determiner *elke* and the noun *man*.

In principle the semantics of *elke* should be $\lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x)$, the semantics of *man* $\lambda y.man(y)$ and the semantics of *slaapt* $\lambda z.sleep(z)$ and application and conversion would go as in (27). (Event semantics is ignored here, as it is not relevant for the mechanism under discussion.)

(27) $\lambda P.\lambda Q\forall x.P(x) \rightarrow Q(x)(\lambda y.man(y))(\lambda z.sleep(z))$
$\Rightarrow$
$\lambda Q.\forall x.\lambda y.man(y)(x) \rightarrow Q(x)(\lambda z.sleep(z))$
$\Rightarrow$
$\lambda Q.\forall x.man(x) \rightarrow Q(x)(\lambda z.sleep(z))$
$\Rightarrow$
$\forall x.man(x) \rightarrow \lambda z.sleep(z)(x)$
$\Rightarrow$
$\forall x.man(x) \rightarrow sleep(x)$

In the implementation in Delilah, there are two things that complicate this picture. One is that nested Cooper-storage is used to handle possible ambiguities (Cooper, 1975; Keller, 1988). Second, as Pereira and Shieber (1987) have shown, Prolog is not very suitable for lambda conversion. Since the parser is written in Prolog, it makes use of partial execution, which is the solution that Pereira and Schieber propose. This means that a lot of the work that is normally done through the application of lambda terms is here done through the binding of variables. A third feature that makes the structure slightly less transparent at first sight is the event semantics for the verb. This is not a requirement of the implementation, but an upgrade of the semantic analysis.

The Prolog version of the semantics of *elke man*, can be represented as $\lambda Q.\forall x.man(x) \rightarrow Q$, and the Prolog version of *slaapt* as *sleep(z)*. This way only one of the terms has a lambda abstraction. In order to get the effect that the second abstraction used to take care of, the semantics of *elke man* (from its position in the store) binds the variable z in the semantics of *slaapt*. It is straightforward that this binding of a variable has the same function as lambda abstraction: it indicates which variable is to be substituted.

The semantics exists of a store and a body. The store is an instance of Cooper-storage (Cooper, 1975). This is where the semantics of the arguments is stored till application. The semantics of *elke*, *man* and *slaapt* are given in (28), (29) and (30) respectively. *Elke* has just one argument, indicated by 'Semarg' in the semantics. In the case of (26), 'Semarg' will be unified with the semantics of *man*. The stores of *man* are empty, as it doesn't take any arguments. Apart from the subject, also the event that is introduced by the verb is in the store. The semantics of the event argument is$\lambda H.\exists I.$ sleep(I) &

event(I) & H. Also the temporal argument, which is treated as a pronoun, is in the store. 'SemSubj' stands for the semantics of the subject, which in this case will be *elke man*.

(28) *semantics of* elke *(every)*

{store:{Semarg *applied to* X *binds* P},
body:λQ. $\forall$X.(P $\rightarrow$ Q)}

(29) *semantics of* man *(man)*

{store:{},
body:λY.man(Y)}

(30) *semantics of* slaapt *(sleeps)*

{store:{SemSubj *binds* S,
λR.$\exists$E. sleep(E) & event(E) & R *binds* V,
temporal_pron *binds* T},
body: AtPres(T).(experiencer_of(V, S) & attime(V, T))}

Below is indicated what these would look like without partial execution.

(31) *elke* 'every'

{store:{SemArg *to be converted against* P},
body:λP.λQ. $\forall$X.(P(X) $\rightarrow$ Q(X))}

(32) *semantics of* man *(man)*

{store:{},
body:λY.man(Y)}

(33) *semantics of* slaapt *(sleeps)*

{store: {SemSubj *to be converted against* S,
λR.$\exists$E. sleep(E) & event(E) & R(E) *to be converted against* V,
temporal_pron *binds* T},
body: λAtPres(T).(experiencer_of(V, S) & attime(V, T))}

After the semantics of *man* is put in the store of *every* to form the SLF(Stored Logical Form) of *every man*, which is then in turn put in the store of *sleeps*, the SLF for the whole sentence is (34).

(34) *SLF of (26)*

{store: {{store: {store: {},
body: λY.man(Y)*applied_to* X *binds* P},
body: λQ.$\forall$X.(P $\rightarrow$ Q) *binds* S},
λR.$\exists$E. sleep(E) & event(E) & R *binds* V,
temporal_pron *binds* T},
body: AtPres(T).(experiencer_of(V, S) & attime(V, T))}

The SLF is a representation that is underspecified for scope of quantificational elements and optional bindings of pronouns. Note that everything comes with its own store and that stores are nested. This means that structures with nested quantifiers are also covered. Next, the stores are applied to arrive at one or more readings for the sentence.

2.3.2 Conversion of lambda terms

This section explains how the stored lambda terms are applied to generate the readings. All conversion operations on lambda terms are performed by three rules, which are each essentially beta-reduction. The need for partial execution prohibits a straightforward application of the lambda calculus.

Application of the stores in (34) is as follows. The temporal pronoun is lexically bound and does not need conversion. It is in the store to check for other binders, since temporal pronouns can bind each other. 'λY.man(Y)' applied to 'X' yields 'man(X)', and since this contains no quantifier and stores are empty the bound variable 'B' can simply be unified with 'man(A)'. This gives us the semantics in (36) for *elke man*, once the stores are applied.

(35) *semantics of* elke man *(every man), before application*

{store: {store:{}
body: λY.man(Y) *applied to* X *binds* P},
body: λQ.$\forall$x.(P $\rightarrow$ Q)}

(36) *applied version of (35)*
λQ.$\forall$X.(man(X) $\rightarrow$ Q)

Suppose the semantics of the event argument in the store is applied before the subject. The lambda term is applied to the body, and the quantifier is going to bind the variable 'V' in the body, with the following result:

(37) {store: {{store: {store: {},
body: λY.man(Y) *applied_to* X *binds* P},
body: λQ.$\forall$X.(P $\rightarrow$ Q) *binds* S}},
body: $\exists$E. sleep(E) & event(E) & AtPres(T).(experiencer_of(T, S)
& attime(E, T))}

Then the semantics of the subject, still in the store, is applied to the new body, with the quantifier binding the variable S in the body. (The store of the subject itself is applied as we have seen earlier.) This yields the following semantic representation for the sentence:

(38) *reading of (26)*
$\forall$X.(man(X) $\rightarrow$ $\exists$E.((sleep(E) & event(E)) &
AtPres(T).(experiencer_of(E, X) & attime(E, T))))

The relative scope of quantifiers depends on the order in which the quantificational elements in the store are applied to the body.

Let us now look at it in more detail. In Delilah, convert/4 takes care of the conversion of lambda terms. The first argument is the stored item under consideration, the second argument is the variable that this stored item binds in the body, the third argument is the body and the fourth argument is the result. The instance of convert/4 shown below applies a stored quantifier to a body, e.g. applying *elke man* to *slaapt*. Note that *elke man*, though syntactically an argument is here the functor.

(39) *double beta reduction, for a stored quantifier applied to a body*

```
convert( λVar.Quant[8], Bound, Body, Quant ) :-
        !,
        binder( Quant, Bound ),
        Var = Body.
```

In applying *elke man* to *slaapt*, the initial values are the following:

λ`Var.Quant`:	λP.∀X.man(X) → P
	(`Var`: P
	`Quant`:∀X.man(X) → P)
`Bound`:	Z
`Body`:	sleep(Z)
`Quant`:	∀X.man(X) → P)

Then, binder/2, given in 40 picks out the variable bound by the quantifier in Quant, i.e. 'X', and unifies it with Bound, i.e. 'Z', which also occurs in the body. This means all 'Z's become 'X's. (`Kwantor^Var^_Scope`: ∀X.man(X) → P, `Var` :X)

(40) `binder(_Kwantor^Var^_Scope, Var)`

Next, 'P', the variable that is abstracted over in the quantifying expression is made equal to the body, which is by now 'sleep(X)'.

So the final values are the following, where the value for Quant, the fourth argument is taken as the result of the operation:

λ`Var.Quant`:	λsleep(X).∀X.man(X) → sleep(X),
	(`Var`: sleep(X)
	`Quant`: ∀X.man(X) → sleep(X))
`Bound`:	X
`Body`:	sleep(X)
`Quant`:	∀X.man(X) → sleep(X))

This corresponds to performing the first conversion step in (41) and finalizing the second conversion step, which had already been partially executed, i.e. the lambda abstraction had already been removed and the variable to be substituted had been indicated through binding. Only the actual substitution still needed to be done.

[8] Adapted for readability. The actual notation is `Var@Quant`, because the usual symbols of the lambda calculus are not available.

(41) λP.$\forall$x.man(x) $\rightarrow$ P(x)(λy.sleep(y))
$\Rightarrow$
$\forall$x.man(x) $\rightarrow$ λy.sleep(y)(x)
$\Rightarrow$
$\forall$x.man(x) $\rightarrow$ sleep(x)

The conversion steps described so far apply to cases where the stored element is the functor and the body is the argument. This is what happens if the stored element is a quantifier. The stored quantifier is applied to the body.

Another preparatory step used in the semantics of Delilah is application to a variable, indicated in the store. This occurs when the body is the functor and the stored element is the argument. So here the syntactic argument is also the semantic argument. An example is the application of *elke* to *man*. Let us first look at the traditional, theoretical way of computing the semantics of *elke man*. This is illustrated in (42), where the semantics of *elke* is $\lambda P.\lambda Q.\forall x.\ P(x) \rightarrow Q(x)$ and the semantics of *man* is $\lambda y.man(y)$.

(42) λP.λQ.$\forall$x. P(x) $\rightarrow$ Q(x)(λy.man(y))
$\Rightarrow$
λQ.$\forall$x. λy.man(y)(x) $\rightarrow$ Q(x)
$\Rightarrow$
λQ.$\forall$x. man(x) $\rightarrow$ Q(x)

The reduction steps in (42) are identical to those in (41). What makes the two situations different, is that here, the functor is the term in the body, as it corresponds to the syntactic head, and the argument of this function coincides with the syntactic argument of the head, whereas, in the case discussed previously, the roles were reversed.

In Delilah, the semantics of *elke* is $\lambda Q.\ \forall X.(P \rightarrow Q)$. It contains only one abstraction. The first argument, P, is not abstracted over, but bound from the store. This is similar to what we have seen above for the subject of *slaapt*. But the situation here is simpler. The semantics of the argument will simply substitute P in the body (as if there was an abstraction over P and the body was applied to the stored item). But first, the semantics of *man* is applied to the variable that is bound by the quantifier in the body. This does the job of the second reduction step in (42). It also solves the abstraction in the argument term right away. In sum, the second reduction step is performed before the first one, while the argument is still stored. The first reduction step is done through the binding of a variable, rather than lambda abstraction.

(43) *semantics of* elke man *(every man)*

{store: {store:{},
body: λ Y.man(Y) *applied to* X *binds* P},
body: λQ. $\forall$X.(P $\rightarrow$ Q)}

The relevant conversion rule is the following, which unifies the variables X and Y and then calls a new instance of convert/4.

(44) *in-store beta reduction*

```
convert( λVar.Quant$Var, Bound, Body, Result ) :-
       convert( Quant, Bound, Body, Result ), !.
```

The rule that applies then, is a simple application rule; the second stage of an already partially executed beta reduction.

(45) *substitution: finalizing beta reduction*
```
convert( Q, Q, Body, Body ).
```

The first Q is *man(X)* (the semantics of *man* that has already been applied to *X*). This is unified with the second Q, which was the variable P (bound), which also occurs in the body. This leads to the result, which is the body with the variable P substituted with *man(X)*. This way P gets its value, without having been abstracted over.

Abstraction is always over something that is syntactically not an argument. Since syntactic arguments are selected, their semantic contribution can be accommodated more directly.[9]

Although direct reference to semantic types is not made anywhere, all applications of lambda terms are compatible with a typed lambda calculus. The anti-symmetry between functions and arguments guarantees this. In other words, it is always clear what is the function and what is the argument.

Due to partial execution no additional solutions are needed, for well-known type-related problems such as applying two quantificational `np`'s of the type $<< e, t >, t >$ to a transitive verb of the type $< e, < e, t >>$. Partial execution lets the transitive verb appear as type t, and the quantifiers as if they were of type $< t, t >$.

In sum, there are two different situations; The stored item (syntactically an argument) can either be the functor, taking the body as a semantic argument, or it can be the argument of the body also semantically. These two cases are handled in two slightly different ways, but they have in common that (relative to standard lambda conversion) the first conversion step has been partly prepared beforehand, and the second step is applied before the first step is finalized.

2.3.3 Stores and quantification

The use of stores is a mechanism for the underspecification of scope. When all arguments are still in store, all scope relations are still unspecified. Such an underspecified representation of a sentence is called Stored Logical Form (SLF), inspired on Alshawi's Quasi Logical Form (Alshawi, 1992). SLF is the result of the unification of the semantic fields in the graphs. It comes about when the entire graphs unify. It is in that sense compositional; the meaning of the whole is determined by the meanings of its parts and the way they are combined. All decisions are taken locally. The semantic composition

[9]The rule of composition has become obsolete, ever since subjects are part of the template from the beginning. What happened was that when both the stored quantifier and the body had an abstraction, the abstraction over the body would be ignored in the application and then added again to the result. This abstraction would be an abstraction over the subject.

```
(1) convert( λVar.Quant, Bound, λX.Body, λX.Quant ) :-
        !,
        binder( Quant, Bound ),
        Var = Body.
```

can be described as functional application of the meaning of one subconstituent to the meaning of the other one. It does not comply with the requirement of stricter versions of compositionality, that the meaning of either subconstituent be opaque to this application (Egg, 2005). In some cases the internal structure of the semantics of the argument is referred to. LF is less compositional, because it is not a local process. It only starts when all unifications have taken place. Readings of a sentence are, however, and fortunately, still predictable from the meaning of the parts and the way they combine, encoded in the constructions and the SLF-to-LF algorithm, which is fully based on information in the sentence graph. It is not the case that recognition of a full sentence is strictly necessary to start the process. The conversion rules can be applied as soon as a constituent has been formed. An applied logical form can be derived for each constituent.

The alternative to SLF is having flexible syntactic types and letting each reading be the result of a different derivation, as happens for example in traditional Categorial Grammar. This leaves no room for underspecification.

The storing mechanism used in SLF has moved away a bit from the original Cooper-storage. The store actually consists of three parts: a quantifier store, a pronoun store and a reflexive store. The semantic field has the following form: {{[QuantifierStore], [PronounStore], [ReflexiveStore]}, Body}. Apart from the semantics of the arguments, the store also contains references to the nodes in the syntactic tree where the arguments are located. As mentioned before, stores are nested, a property that was first introduced by Keller (1988).

From SLF to LF: getting the readings

Here I discuss the mechanism that derives LF readings from the SLF. It is called apply_store. The apply_store algorithm spells out the different readings based on the SLF. What it does, is to re-order the elements in a store and decide which elements can be promoted to a higher store. This decision is based on grammatical properties encoded in the sentence graph. Then, for each order obtained in this process, it applies functions to arguments according to a small set of conversion rules, as explained above. Apply_store, thus, does not add anything new to the semantics. It manipulates the order of stored elements and then takes care of the final function application.

In Delilah, the scopal order of quantifiers can be influenced at several points. First of all, arguments are put in the store in a certain lexically determined order. The design is that the order in which arguments of the same lexical head appear in the store, should not matter. But if no additional operations are carried out on the store before it is applied, the elements will be applied in that order. Delilah provides several mechanisms to reorganize the store. Stored items are split up in scopal and non-scopal elements. Some grammars let all DPs introduce scopal ambiguity, but for proper names, for example, this is certainly not necessary. By putting them in the non-scopal list, they are kept out of scopal interactions. The scopal elements that do relevantly contribute to ambiguity are then reshuffled in order to generate more readings by backtracking. These scopal elements are then also available for raising to a higher level, provided they are not inside an island. Since every item in the store comes with a reference to the position in the syntactic tree it comes from, the algorithm is able to check for islands. Thus the order of

quantifiers can be influenced at three points:

1. in the lexicon, by the order in which they are put in the store;

2. in the processing of the store, by distinguishing between different types of stored items and by reshuffling

3. by raising to a higher store under certain conditions

The drawback of the algorithm is that quantifiers that have been raised to a higher store will be re-ordered there in all possible orders again. This way the same reading can be generated more than once.

When spelling out the different readings, redundancy should be avoided as much as possible. This means first of all, not generating the same reading twice. Secondly, it would be good to not generate readings that are logically equivalent to previously generated readings. Besides that, one should implement all known restrictions on which readings are actually possible.

Let me first sketch a strategy that does not generate the same reading more than once. Instead of computing a new reading after each permutation or move we can use an approach, inspired on the copy theory of movement (Chomsky, 1993), where all quantifiers that can move to higher stores, are first copied to all the stores that they can move to. The quantifiers can remain unordered with respect to each other within a store. Then, a procedural algorithm determines the different possible orders which contain exactly one copy of each quantifier. For example, if there only is one copy of a particular quantifier, and this copy occurs in the highest store, then this quantifier can only be preceded by other quantifiers that have a copy in the highest store.

The algorithm may work as follows: First, take the first order you find starting at the beginning; start with the first quantifier you encounter, then the next, skipping quantifiers you already have. Of this initial order, try if you can swap the last two quantifiers. Then try to find an alternative for the second last, et cetera, till you exhaust all alternatives. Two quantifiers can be swapped if they are in the same store, or if a copy of the first also occurs in a lower store than the second. This way you never get two identical readings.

So far, I have only discussed the scope of generalized quantifiers in relation to other generalized quantifiers. There may be cases, however in which it matters in which store a quantifier is realized, even if the order with respect to all other quantifiers is in both cases the same. This can happen when there is intervening material that does not occur in the store, but does have scope effects, such as modals and scopal adverbials. Such intervening material is not in the store, but will be fixed in a syntactic position between two stores, and needs to be checked for if its scope effects are to be accounted for.

The classical example is (46), where *a unicorn* can either scope below or above *seems*.

(46) A unicorn seems to be in the garden.

To get this right at all in Cooper-store, which does not allow for quantifier lowering, a raising analysis is needed. *A unicorn* starts out, semantically, as an argument of *be*. From

there it can raise to the higher store, the store of *seems*. Because of this intervening modal, it matters in which store the quantifier is realized, even if there is only one quantifier.

So for each quantifier order it should be checked whether one of the quantifiers can also be realized on the other side of a fixed scopal element, while maintaining the same position with respect to the other quantifiers. This would be an extra reading.

This approach actually gives us three levels of LF: the SLF that is the direct result of the derivation, the underspecified representation with copies of the quantifiers in all possible stores, and the fully specified readings.

Equivalent readings

The procedure described above prevents generating the same reading twice. In some cases, however, it still derives two or more readings that are equivalent to each other. This may be (partly) prevented by imposing additional conditions on permutations, such as: two quantifiers that are of the same type, (i.e. two universals, or two existentials) with nothing intervening should not be swapped with each other. (Here the quantifier is considered to be the whole DP, see Barwise and Cooper (1981).) Koller and Thater (2006) wrote a redundancy elimination algorithm for underspecified descriptions. It distinguishes classes of readings, the members of which are equivalent to each other, but maintains underspecification. The algorithm works on dominance graphs. It should be possible to translate Delilah's nested Cooper-stores into dominance graphs, because all the information needed seems to be available. The body and the stored elements can function as fragments and the binding of variables between fragments is known. After it has been calculated to what stores the quantifiers can move, it will be clear if one quantifier has to have scope over another. This information can then also be used. I will therefore assume that Koller and Thater's algorithm can be used for Delilah.

Restrictions on scope

Some island constraints have been implemented to prevent unwanted readings. Islands are subdivided into the following types: decreasing, wh, factive, intensional, and lexical. Island constraints are general syntactic constraints. Some extra constraints may apply to the behavior of particular types of quantifiers, though.

The sentence in (47) has often be claimed to have an enormous amount of readings.

(47) A politician can fool most voters on most issues most of the time, but no politician can fool all voters on every single issue all of the time.

But how many (different) readings does it really have? Can we exclude any readings without applying world knowledge?

First of all, it seems to be impossible, or at least very hard, to give the universals in the second conjunct wider scope than 'no politician'. Even the simpler (48) cannot mean that all voters are such that no politician can fool them. When 'all' is replaced with 'most', however the inverse reading is available.

(48) No politician can fool all voters.

The phenomenon that readings in which a universal is raised over negation are disfavored can also be observed in the contrast between (49a) and (49b). Whereas (49a) easily gets the non-absurd reading where *every corner* scopes higher than *a policeman*, in (49b) it can only get narrow scope, yielding a trivial reading.

(49) a. A policeman is standing on every corner.
b. No policeman is standing on every corner.

Another thing is that, although it is possible to give all three 'most's together scope over 'a politician' (if 'a politician is interpreted as an existential, not if it's a generic), it does not seem to be possible to let only one of the 'most's take wide scope over the existential, while leaving the others behind. '*Most*'s, and universal quantifiers, seem to cluster. This is also illustrated by the following simpler examples. Whereas (50a) appears to have all readings that we would expect, (50b) does not seem to be able to mean that for every book there was a professor who recommended it to every student, or that for every student there was a professor who recommended him every book.

(50) a. Every professor recommended a book to a student.
b. A professor recommended every book to every student

It is not clear whether this is a hard fact nor what causes it.

Permuting the order of the universals in the second conjunct does not yield any new readings unless there are reasons to multiply the sets under consideration. The situation for the 'most's in the first conjunct is more complicated, but at least it is very difficult to intuitively distinguish different readings, even in a simpler example.

(51) a. Most politicians can fool most voters.
b. Most voters can be fooled by most politicians.

Intuitively, (51a) and (51b) seem to entail each other. One may argue, that this is because they are both ambiguous in the same way. That would predict that (52a) and (52b) are just as easily accepted as equivalents. Here however it is clearly felt that the equivalence is dependent on the choice of reading that is made for each of the sentences.

(52) a. Some politician can fool all voters.
b. All voters can be fooled by some politician.

Moreover, it is tempting to judge even the unambiguous sentences (53a) and (53b) as equivalent. Most people need pen and paper to convince themselves that logically they are not.

(53) a. For most politicians it is the case that they can fool most voters.
b. For most voters it is the case that they can be fooled by most politicians.

In the light of the work of Anderson (2004), who argues on the basis of psycholinguistic experiments, that human interpreters have a preference for the surface-scope reading, because that is the syntactically simplest reading, it is likely that humans only compute one reading for the in principle ambiguous most-most sentences. The chance that they

have good reasons to assign the inverse-scope is very small, since they can barely tell the difference between the default reading and its alternatives. For a parser, it would therefore be quite safe to also compute only the surface-scope reading.

One thing is clear: further research is necessary on restrictions on quantifier scope. All of the issues discussed in this section require more work.

2.3.4 Other scopal elements

Considering unstored scopal elements as fixed, means that you cannot underspecify scope relations of these fixed elements with respect to each other. Ambiguity in such scope relations will have to be considered syntactic ambiguity. In order to be able to underspecify the scope relations between scopal elements, these elements have to be in the store. It is therefore worth investigating whether it is possible to store all scopal elements, i.e. not only quantifiers, but also modals and adjuncts. I will first discuss whether this is technically possible in the format of Delilah, and then whether, or to what extent, it is desirable.

Storing operators

Let us look at the ambiguity in (54). The sentence can mean either that it is often possible for Alice to come or that it is possible for Alice to often come.

(54) Alice kan vaak komen.
Alice can often come
'Alice can come often.'

At present, *kunnen* and *vaak* are represented as operators. That is, the semantics of *kunnen komen* ('to be able to come') has a schematic structure as in (55a) and the semantics of *vaak komen* ('to come often') has a schematic structure as in (55b).

(55) a. *kunnen komen*
possible(∃e & come(e) . . .)
b. *vaak komen*
often(∃e & come(e) . . .)

The semantics of (epistemic) *kunnen* is (56). The result of applying the semantics of the VP complement to X and then adding a new abstraction over X in the body, is that the modal operator is inserted under the abstraction. This abstraction is in practice the abstraction over tense, to be resolved by the tense of the finite verb. The quantifiers introduced by the arguments of the complement verb (included in SemCompl) end up in the scope of the modal, if they are not raised to a higher store.

(56) {store: {SemCompl *applied to* X *binds* A},
body: λX.possible(A)}

The simplest way of putting the operator in the store is (57), which after application will have the same result as (56).

(57) store: SemCompl *applied to* X *binds* A, λY.possible(Y) *binds* B
body: λX.B(A)

This however does not help, as it still does not allow for scope ambiguity, as can be seen below, where *vaak* takes an argument headed by *kunnen*.

(58) *vaak kunnen VP*
{store: {store:{SemVP *applied to* X *binds* A, λY.possible(Y) *binds* B},
body: λX.B(A) *applied to* W *binds* C,
λZ.often(Z) *binds* D},
body: λW.D(C)}

Whereas a quantifier binds a variable in the body, an operator substitutes its variable. Therefore, the only reading that can be obtained from this SLF is the one where *possible* scopes over *often*, i.e. *possible(often(∃e & come(e) ...))*. Only a different syntactic analysis, where *vaak* is in the complement of *kunnen* gets us the other reading.

This shows that storing cannot usefully be applied to scopal operators in exactly the same way as to quantifiers.

An alternative is to not let operators in the store bind a variable in the body, but marking them for being operators, for example, by stating that they bind 'nothing'. The store applying algorithm should then apply the operators to the body, when it is their turn. This way they can be applied to the body in different orders, yielding different results.

A way of implementing this is by adding an instance of convert/4, that is reminiscent of the old composition rule, but without a variable being bound in the body.[10]

(59)
```
convert( λVar.Quant, nothing, λX.Body, λX.Quant ) :-
        !,
        Var = Body.
```

This would resolve a structure like the following:

(60) store: {store: {SemVP *binds* A, λY. possible(Y) *binds* nothing}
body: A *binds* B,
λZ.often(Z) *binds* nothing}
body: B

[10] Variable binding in the body when the body consists of only that variable is a bit strange. We can in principle replace the body with the body of the argument right away. The semantics in the template of a scopal operator would then be constructed as follows:

(1) sem:
store: StoresArg, λY. Operator(Y)
body: BodyArg

semarg:
store: StoresArg
body: BodyArg

semhead: Operator

A disadvantage is that this makes it more difficult to restrict the scope of particular items. Especially for the arguments of the VP complement this may be problematic.

If *λY. possible(Y)* is raised to a higher store, namely the same one where *λZ.often(Z)* is in, the two operators can be applied in either order. The arguments of the VP complement, here included in *SemVP*, can in principle also be raised to a higher store and take any kind of scope, if they are not prevented from doing so.

Now that I have shown that it is possible to store scopal elements other than quantifiers, and underspecify for their scope I will discuss in which cases it is useful to do so.

Split scope effects

Before I discuss the problematic aspects of the examples used above, I will first present a case in which it is relatively clear that the strategy described is necessary and successful.

Rullmann (1995) and Penka and Zeijlstra (2005) describe split scope effects for negative indefinites, such as Dutch *geen* ('no') (see also Jacobs (1980), Kratzer (1995), de Swart (1996)). They observe that the scope of a modal can intervene between the scope of the negative operator and that of the existential quantifier, as is illustrated in the (a) readings of (61) and (62). The (a) reading is in both cases the most natural one and in (62) even the only really sensible one, since the (c) reading is not available under intensional verbs and the (b) reading is trivially true in any world where there are no unicorns.

(61) Ze mogen geen verpleegkundige ontslaan.
they may no nurse fire

a. 'They are not allowed to fire any nurse. $\neg > may > \exists$
b. 'There is no nurse who they are allowed to fire.' $\neg > \exists > may$
c. 'They are allowed not to fire a nurse.' $may > \neg > \exists$

(62) Hans zoekt geen eenhoorn.
Hans seeks no unicorn

a. 'Hans is not trying to find a unicorn.' $\neg > seek > \exists$
b. 'There is no unicorn Hans is trying to find.' $\neg > \exists > seek$
c.* 'Hans is trying not to find a unicorn.' $seek > \neg > \exists$

It is therefore argued that negative indefinites should be considered as complex lexical items consisting of a negative and an indefinite part. Potts (2000) shows under what conditions even English *no* shows such effect.

In Delilah the semantics of *geen* is now built up as in (63). Here, it is not possible to generate a split scope reading. The negation and the existential will always be together. (SemN stands for the semantics of the nominal argument of the determiner.)

(63) *semantics of* geen
store:SemN *applied to* X *binds* P
body:λZ.¬∃X.P & Z

If we put the negative operator in the store, however, as in (64), split scope readings will be derivable, by raising the negative operator to a higher store.

(64) *proposed semantics of* geen
store:SemN *applied to* X *binds* P, λY.$\neg$(Y) *binds* nothing
body:λZ.$\exists$X.P & Z

This exactly predicts the readings we get for (61). It also predicts that the existential will never scope higher than the negation. In order for the existential to get scope over the negation, the negation operator would first have to raise to a higher store, after which the remaining part of the semantics of *geen N* would need to raise to a position where it can outscope it. This would amount to remnant movement, which Delilah does not permit. Introducing remnant movement in Delilah is undesirable, since it would largely, if not entirely, undo the effect of nested stores (Keller, 1988).

For (62) I need to assume that *zoeken* 'seek' has to be analyzed as 'try to find' (where 'try' introduces the intensionality). Why the (c) reading is unavailable remains unexplained. Penka and Zeijlstra suggest it may be for pragmatic reasons. My guess would be that it is an effect of incorporation, since (65) does have all three readings.

(65) Hans probeert geen eenhoorn te vinden.
Hans tries no unicorn to find

a. 'Hans is not trying to find a unicorn.' $\neg > try > \exists$
b. 'There is no unicorn Hans is trying to find.' $\neg > \exists > try$
c. 'Hans is trying not to find a unicorn.' $try > \neg > \exists$

This particular effect, which is specific to verbs with intensional objects, will be hard to implement, but for the rest, the approach of putting the negative operator in the store seems very fruitful for split scope effects. It will also prove very useful for the analysis of *geen* in collocations, which I will discuss in section 2.3.9. Note also, that there is no way to analyze simple negation as a quantifier, quantifying, for example, over times or possible worlds, although for many other operators that may be an option. Simple negation can only be an operator. I therefore conclude that at least for these negation cases, the operator-in-store technique is required.

Modal verbs and scopal adverbs

The operator-in store analysis for the original ambiguous examples in section (2.3.4) is more problematic, because the ambiguity in these examples is normally considered syntactic. Two different syntactic trees accidentally have the same surface order. Although for computational semanticists it would of course be convenient to be able to underspecify for this kind of ambiguity, getting two different syntactic analysis for a sentence like (54), seems unavoidable, since, no matter your analysis of adverbs, *vaak* will always be able to combine with *komen* as well as with *kunnen komen*. Examples (66a) to (66c) show that *vaak* can combine with matrix verbs as well as with embedded verbs, depending on its position.

(66) a. Vaak ontkent hij te zijn gekomen.
often denies he to have come
'Often he denies to have come.'

b. Hij ontkent vaak te zijn gekomen.
he denies often to have come
'He often denies to have come.'/ 'He denies to often have come.'

c. Hij ontkent dat hij vaak is gekomen.
he denies that he often has come
'He denies that he has often come.'

In addition, the scope of adverbs amongst each other is mostly fixed, and determined by the order. (67a) and (67b) are unambiguous. Likewise, the relative scope of modal verbs is completely determined by their order.

(67) a. Bob komt niet vaak.
Bob comes not often
'Bob does not come often.'

b. Bob komt vaak niet.
Bob comes often not
'Bob often doesn't come.'

Where a sentence with two adverbials does display an ambiguity which might be considered scopal, such as (68) from Broekhuis (1999), this still seems to originate from a syntactic ambiguity. Broekhuis argues that in one reading *vaak* is interpreted as a VP-adverb and in the other reading as a clausal adverb.

(68) Alice gaat op zondag vaak naar de kerk.
Alice goes on Sunday often to the church
'Alice often goes to church on Sundays.'

I conclude that the scope of modal verbs and adverbials is fixed by their (surface) syntactic position. In the next section I will discuss whether that excludes underspecification.

Syntactic vs. semantic ambiguity

The scope ambiguity of quantifiers that is underspecified is considered semantic ambiguity. The difference between syntactic (structural) and semantic (scopal) ambiguity is the result of giving up on a 1:1 relation between syntax and semantics. In theoretical syntax, a sentence that is ambiguous only in quantifier scope still gets two different syntactic trees. These two trees only differ in which copy of which quantifier is interpreted, to put it the minimalist way. Or in more traditional terms, the difference between the two structures is only the result of movement on LF. It is a particular kind of structural ambiguity, but it can still be considered structural.

In computational semantics, on the other hand, underspecification of scope ambiguities is widely embraced.

Also, I concluded that ambiguity in adverb scope is syntactic, whereas a negation that is expressed on a determiner should be stored. This gives us the weird result that the scope of the negation in (69a) en (69b) is determined by their syntactic position,

i.e. the negation occupies different positions, whereas the scope of the negation in (70a) and (70b) would not be determined by the syntax, i.e. the negation occupies the same syntactic position in both cases.

(69) a. Ik kan niet komen. (*normal intonation*)
I can not come
'I can't come.' ($\neg > can$)

b. Ik kan (ook) NIET komen.
I can (also) NOT come
'I can also NOT come. ($can > \neg$)

(70) a. Ik kan geen soep bestellen. (*normal intonation*)
I can no soup order
'I can't order soup.' ($\neg > can$)

b. Ik kan (ook) GEEN soep bestellen.
I can (also) NO soup order
'I can (also) NOT order soup.' ($can > \neg$)

However, the cases seem completely parallel, except for the fact that (70a) and (70b) contain an indefinite direct object. An alternative analysis that has been proposed (Klima, 1964; Rullmann, 1995), and seems more likely, is that the negation in (70a) and (70b) is only pronounced on the determiner, but that its actual syntactic position is a different one. The simplest assumption would be that the negation in (70a) is in exactly the same position as in (69a), and that the negation in (70b) is in the same position as the negation in (69b). Nevertheless, the approach sketched above for negative indefinites appears to be the most suitable one for implementation in Delilah.

The difference in treatment of the two cases is then an artefact of Delilah's rather lexicalist and surface oriented syntax. It may be the case that the whole distinction between syntactic and semantic ambiguity (or at least its significance) is the result of a choice for a not too complex, surface oriented syntax. This choice however, has the advantage that it limits the number of trees that are possible for a sentence.

Allowing that an ambiguous sentence can have two different parse trees and one unified underspecified semantic representation is problematic, because it would suggest that each of the trees can have any of the readings, which is not correct (Bunt, 2007). It is an open question whether there can be semantic underspecification based on syntactic underspecification.

2.3.5 Scope disambiguation and underspecification

The assumption that scopal elements can in principle take scope in any order creates a problem of massive ambiguity. A sentence with *n* scopal elements potentially has *n*! readings.

To avoid having to enumerate all readings, several underspecified representation formalisms have been introduced. Cooper-storage is one of them. Other prominent ones are MRS (Copestake et al., 2005), dominance diamonds (Egg, 2004) and hole

semantics (Bos, 1996). It is not too difficult to represent all $n!$ readings of a sentence with n scopal elements in a compact way. Difficulties arise if one wants to be able to represent that certain readings are excluded. Ebert (2005) shows that no underspecified representation formalism can be both expressively complete and avoid combinatorial explosion. Expressively complete means that in all cases where some readings are ruled out, for example by partial disambiguation through context, the formalism is able to represent all remaining readings in one underspecified representation. In order to do that it would have to be able to represent each subset of the set of n! readings, which means $2^{n!}$ different sets.

It looks like underspecification does not solve all our problems. Of course having a reliable algorithm to obtain the intended meaning of a sentence would be preferable[11]. Unfortunately such an algorithm has not been invented yet. Here I will discuss two alternatives to the underspecification approach described above. The first one is to only give the most general reading. The second is to initially compute only the most likely reading.

An interesting way of avoiding computing and enumerating all possible readings, is to only give the most general reading. For example, the sentence in (71), in its surface scope reading, that is the reading with wide scope for *every*, is also true in a situation where all girls push the same truck.

(71) Every girl pushed some truck

Pietroski and Hornstein (2002), would even go as far as saying that a sentence like (71) is not ambiguous. It only has one reading, which is the wide scope reading for *every*. It can be made true by several different situations, including ones in which all girls push the same truck. An important argument is that in (72) through (74), where an inverse reading would not entail the surface reading, the inverse reading does not appear to be available.

(72) Two girls pushed few trucks.

(73) Few/Most/Several girls pushed no truck.

(74) No girl pushed no truck.

That is, (72) cannot mean that there were few trucks pushed by two girls, (73) cannot mean that there was no truck that few/most/several girls pushed, and (74) cannot mean that there was no truck that no girl pushed. (These facts seem to be related to what was presented in (48).)

For a sentence like (71), one could thus quite safely give only the reading with a wide scope for the universal quantifier. Some possible situations that would make the sentence true may be incompatible with the context. So, inferences that are based not only on this one sentence, but also on the context, should then turn out correct. This means that for representations that aim to support inferencing, this may be an interesting way of underspecification.

[11] I assume that in most cases there is an intended reading, although it is of course true that some utterances may be intended to be ambiguous.

This strategy is less straightforward for examples like (75), though. Here the object can take scope over the subject.

(75) Some girl pushed every truck.

According to Pietroski and Hornstein, sentence (75), in contrast with the sentences (71) through (74), is ambiguous. In their account, they distinguish between strong and weak determiners/quantifiers. When a sentence contains two quantifiers Q1 and Q2 (where Q1 is the subject and Q2 is the object), the LF will look as follows:

(76) [Q1 Q2 [$_{VP}$ Q1 V Q2]]

There are two copies of each quantifier. One is the original copy inside the VP and the other one is the result of quantifier raising. What determines the scope ordering, is which copies are realized and which are deleted. The VP-internal copy of a strong quantifier must always be deleted. This means that if Q1 is a strong quantifier, Q1 will always outscope Q2.

In spite of this, one could of course still consider giving (75) only the most general reading as a means of underspecification (i.e. the reading with wide scope for the universal quantifier). This approach would lead to an algorithm where the order of the quantifiers in the underspecified reading would depend on the types of quantifiers involved. For example, a universal quantifier would always have scope over an existential one.

The problem with this approach is that not for all quantifier pairs one order entails the other. Consider (77).

(77) No girl pushed two trucks

The sentence is ambiguous. But if there was no girl who pushed two trucks, this does not necessarily mean that two trucks were not pushed. Maybe each girl pushed just one truck, but all trucks got pushed by a girl. And if there were two trucks that were pushed by no girl, that does not have to mean that there was no girl who pushed two trucks. So in this case there is no most general reading, of which the other reading is a special case. In conclusion, although the strategy of giving the most general reading seemed interesting, it fails because there is not always a most general reading.

Pietroski and Hornstein's approach can possibly be used to restrict the number of readings. It does not cover all cases, though. Whereas it predicts, for instance, the ambiguity of (77), since both quantifiers are weak, it also predicts (72) and (74) to be ambiguous for the same reason, which they are not.

A different approach is to initially give only the most likely reading and only compute alternative readings if the first turned out not to be compatible with the context or with world knowledge. Computing an alternative reading should be guided by context, and be conservative. One should not invert more than what is necessary to make the sentence compatible with its context.

Anderson (2004) shows psychological evidence that surface scope readings are more easily computed by humans than inverse scope readings, even in contexts that favor the latter. Even in unambiguous inverse-scope sentences, the greater processing cost can be

measured in terms of slower reading times. (Also the examples (72) through (74) seem to indicate that surface order is important.) The problem is that Anderson's experiments are limited to active subject-verb-object sentences containing the quantifiers *a* and *every*. Kurtzman and MacDonald (1993) show that in passive sentences there is a much weaker preference and in complex NPs containing two quantifiers, the most embedded one is preferred to have the widest scope. It seems it is not always clear what makes something the simplest reading. This makes it difficult to systematically predict which will be the simplest reading.

Several factors play a role in how people pick a reading for a multiply quantified sentence, among which at least the following:

1. syntax
   Effects of syntactic configurations, such as islands, can completely exclude certain readings. Syntax can also influence preference for one reading or the other, as Anderson shows. Readings that are syntactically more complex seem to be dispreferred.

2. linear order
   Linear order is likely to play a role, because this is the order in which hearers perceive the different quantifiers and presumably also the order in which they process them. As a condition, it is hard to tease apart from syntax and topicality (Bunt, 1985; Fodor, 1982; Johnson-Laird, 1969; Kroch, 1979; Lakoff, 1971; VanLehn, 1978).

3. topicality/discourse
   Topics tend to preferably have wide scope. (They tend to be fronted, and in spoken language, topicality is also marked by intonation.)(Kempson and Cormack, 1981; May, 1985)

4. context
   Context can help to disambiguate, for example by suggesting that a phrase that starts with *a* introduces one or more than one entity (Anderson, 2004).

5. real-world knowledge
   Expectancies based on world-knowledge play an important role. If world knowledge only allows one reading, this probably overrules most other factors, except for hard syntactic constraints or a context that tells the reader that the text is about a world that differs at the relevant point from the world he knows.

6. lexical preference of a quantifier
   For example, *each* has a stronger preference for wide scope than *every* (Beghelli and Stowell, 1997).

7. thematic role effects
   Agents having a stronger preference for wide scope than e.g. experiencers (Grimshaw, 1990; Jackendoff, 1972). (This can also be considered part of syntax.)

In Anderson's experiments it turns out that a context that was intended to disambiguate a sentence, did not always have that effect. This suggest that humans may not always choose the reading that was intended by the speaker. The conclusion is that at this point too little is known about preferred readings and disambiguation in context, to base a general computational strategy on.

There is one other research direction in scope disambiguation worth mentioning. Saba and Corriveau (2001) describe a strategy of disambiguating based on a database that contains knowledge about what relations are normally one to many, many to many, etc. It will for example contain the information that a house is normally on one street, but that there are typically multiple houses on one and the same street. This is interesting, because possibly this kind of knowledge can be harvested, by extracting unambiguous sentences with particular subject-verb-object combinations. (e.g. sentences containing demonstratives).

2.3.6 Adjuncts

Adverbials are well known for being problematic syntactically as well as semantically (Austin et al., 2004). Their syntactic position is not clear. Both syntactically and semantically there appear to be different classes of adjuncts with different behavior, but there is no consensus on the exact classification. Syntactically, adverbials are optional. The question is whether they optional in the semantics too, or fill an existing position, which in absence of an appropriate adjunct would get a default value.

I will talk here about 'adjuncts', because that is what adverbials are now considered to be in Delilah, though the alternative approach that I propose here, may be closer to the "adverbials are specifiers" position (Alexiadou, 1997; Cinque, 2004; Laenzlinger, 1998).

The category of adjuncts

In the present version of Delilah, adjuncts are categorially treated as automorphisms. This means, that they combine with a sentence to form a sentence, or combine with a VP to form a VP, or combine with an NP to form an NP, etc. We can thus say, that they are of the type x/x (or x\x), taking something of category x (on the left or the right side) to form something of the category x. A grammar rule that combines a phrase with an adjunct will have the general form in (78).

(78) $\quad x \quad x\backslash x \Rightarrow x$

This means the adjunct is the functor and the other phrase is the argument. The result of the application is again of category x, and can combine with yet another adjunct in exactly the same way. This accounts for the observation that adjuncts are never required and that another adjunct can always be added. Placement of adjuncts is regulated by modes[12] (explained in section 2.1).

[12]The mode that is used for adjuncts is the same one that makes the third construction possible in verbal clustering.

However, long extraction of adjuncts cannot be handled by this approach (Cremers, 2002). For example, *waar* in (79a) can at present not be interpreted by Delilah as being extracted from the embedded clause.

(79) a. Waar denk jij dat hij mij wou onderbrengen?
where think you that he me wanted lodge
'Where do you think he wanted to lodge me?'

b. Wie denk jij dat hij hier wou onderbrengen?
who think you that he here wanted lodge
'Who do you think he wanted to lodge here?'

Recall from the grammar section that *wie* introduces two grammatical types. The first one is `q/s` and the second one `np`, together with the `np^wh\s`, the whole thing nicely goes to `q`. And since the `np` that the sentence was still searching to the left, was the object of the embedded verb, the question word is interpreted as being extracted from the embedded clause. Now consider *waar*, also introducing a type `q/s`, but next to that an adjunct of the type `x/x`, which in this case will turn out to be `s/s`, combining with a complete `s`. The result is still `q`, but the adjunct will be interpreted as modifying the matrix sentence, i.e. asking about the place of the thinking, rather than as extracted from the embedded clause, which would be the most natural reading.

(80) a. Waar denk jij dat hij mij wou onderbrengen?
[]\q/s []\s/s []\s/[]

b. Wie denk jij dat hij hier wou onderbrengen?
[]\q/s []\np/[] np^wh\s/[]

This is because nothing is looking for an adjunct. The embedded clause is perfectly happy without it. In (79b) on the other hand, there is no problem. Because *wie* is an argument, it is searched for by the verb in the embedded clause, which still wants an object, and therefore it is interpreted as extracted from the embedded clause.

Note also, that this is not a problem for real sentential adjuncts. (81a) does not have a reading where *misschien* ('maybe') is interpreted as modifying only the embedded clause (i.e. (81a) does not have a reading in which it means the same as (81b)). Sentential adjuncts cannot be targeted by *wh*-question words either. (i.e. *misschien* cannot be the answer to a wh-question).

(81) a. Misschien denkt Bob dat Piet komt.
maybe thinks Bob that Piet comes
'Maybe Bob thinks Piet is coming.'

b. Bob denkt dat Piet misschien komt.
Bob thinks that Piet maybe comes
'Bob thinks Piet may be coming.'

NP-adjuncts, such as relative clauses (82a) and prepositional phrases (82b), are not available for long extraction either, (or even short extraction, since no extraction is possible from a complex NP), but they can be right dislocated, giving rise to the same kind of problem.

(82) a. Ik heb de man proberen op te bellen *die de prins beledigd heeft*.
I have the man tried up to phone who the prince offended has
'I have tried to phone the man who offended the prince.'

b. Bob heeft gisteren een boek gekocht *met veel plaatjes*.
Bob has yesterday a book bought with many pictures
'Bob bought a book with many pictures yesterday.'

Treating VP-adjuncts as arguments would solve the extraction problems. Adjuncts can for example be selected as optional arguments. In categorial grammar, that would mean, having a rule like (83) instead of (78) for VPs that take adjuncts, where `x/a` is the type of a VP that optionally takes an adjunct. There is an optional argument slot for an adjunct and if it is filled, a new such slot is created.

(83) x/a a ⇒ x/a

And in addition, one would need (84).

(84) x/a ⇒ x

(84) takes care of the optionality of the adjunctive argument. It lets `x/a` go to `x` even without a positive occurrence of `a`. (83) makes this optionality recursive, allowing for an amount of adjuncts that is in principle unlimited. In other words, the two rules together let a category absorb any amount of adjacent adjuncts and in the end always go to `x`.

An alternative to having (84) is to list everything of type `x/a` also as type `x` in the lexicon for the case where there are no adjuncts and then use normal cancelation for the last adjunct. Then we would use (83) and (85), plus ambiguity in the lexicon. The chartparser is capable of deciding which rule should be used when, to let the whole string go to `s`.

(85) x/a a ⇒ x

Unlike (78), which was an innocent case of cancelation of a negative occurrence of a category against a positive one, (83) and (84) are not the kind of rules that fit in a normal categorial grammar. In fact, (83) is a non-canceling rule.

Bouma and van Noord (1994) avoid this problem, by letting the adjunctive argument be introduced by a recursive lexical rule (or constraint). In the architecture of Delilah, such a lexical rule would be problematic, because Delilah works with the whole lexicon already compiled and indexed in such a way that it can be searched in in linear time. A recursive lexical rule would lead to an infinite lexicon, which therefore can not be created and indexed as a whole. What Bouma and Van Noord do, is to postpone the application of the lexical rule. Only when it is clear what the category of the verb should be to make the sentence grammatical, it is checked whether it can get this category through the lexical rule. This is equivalent to having the kind of syntactic rule in (86). By making it a lexical constraint, they restrict which categories can play the role of `x` in this rule.

(86) x ⇒ x/a

This rule is possibly even worse in a grammar than the previous ones, because, rather than reducing complexity, it introduces extra complexity. It is a sort of lifting rule, introducing an extra slash. This causes the risk that the parse will not terminate. Bouma and Van Noord argue that with a special parsing strategy this can be kept under control. It is, however, one of the advantages of CCG that it can do without type lifting. Therefore this strategy is not very attractive for Delilah. Rules (83) and (84) at least appear less dangerous than (86). (84) is rather harmless, because it gets rid of a slash anyway. (83) is slightly more risky, because it does not reduce complexity, but it does not introduce any new slashes either. And at the point that no more adjuncts are found, `x/a` will reduce to `x`, through (84). It's the running out of adjuncts that terminates the process. (84) and (83) could of course never be general rules in a classical CG. That is also the reason why Bouma and Van Noord make theirs a lexical rule. In CCG, however, rules are not general. Conditions on the cases in which a rule applies can be formulated in the grammar by means of the modes. And in a way the adjunct in (83) is still an automorphism, as it combines with something of the category `x/a` to form something of the category `x/a`.

If one thinks of the slash as a (direction sensitive) division and of composition as multiplication, then one could think of `a` as the number 1. This would make all the rules valid. (84) then corresponds to x/1=1, (83) to x/1·1=x/1 and even x/x=1. Also (86) fits in this picture (x=x/1), but is less suitable for Delilah, as it is a type changing rule. So, whatever approach we take to adjuncts, their grammatical type always corresponds to 1.

If it could be shown that an event only has one slot for each type of adjunct, e.g. time, location, instrument etc. that would make the number of possible adjuncts per verb finite. In this case it might in principle be possible to arrange the optionality in the lexicon, without making it infinite. This is probably not a good idea, though, as the amount of lexical entries will not get infinite, but still extremely large, depending of course on how many different adjunct slots there are and in how many different configurations they can occur. It will still be a lexical explosion. Therefore a plug-in syntax for adjuncts through a non-cancelation rule seems to be the best option for a lexically based system like Delilah[13].

The relatively free position of adjuncts in the sentence, illustrated by (87), was nicely arranged by the mode of composition, when adjuncts were of the category `x/x`.

(87) a. Ik probeer Bob vrijwillig het boek voor Agnes te laten kopen.
I try Bob voluntarily the boek for Agnes to let buy
'I try to let Bob voluntarily buy the book for Agnes.'

b. Ik probeer Bob het boek vrijwillig voor Agnes te laten kopen.
I try Bob the boek voluntarily for Agnes to let buy
'I try to let Bob buy the book voluntarily for Agnes.'

c. Ik probeer Bob het boek voor Agnes vrijwillig te laten kopen.
I try Bob the boek for Agnes voluntarily to let buy
'I try to let Bob buy the book for Agnes voluntarily.'

[13] Syntax based systems have the option of using a fixed syntactic hierarchy for adjuncts, such as the one Cinque (1995) argued for.

Similar results can be obtained with the proposed adjunct rule if the position of the adjunct in the list (i.e. the search agenda) does not matter. The rule should apply when the adjunct is somewhere in the list, not only when it is on top of the list. In order not to block canceling of normal arguments, the adjuncts should either be transparent/skippable or be put at the end of the list. The latter is the most simple solution. Of course, when in composition two agendas of two categories are combined, all adjuncts need to end up in the end again. This boils down to having a separate, unordered, list for adjuncts.

It may become problematic, though, to control for the relative order of adjuncts, if there are restrictions on that (e.g. 'place before time' in English), or to implement restrictions on the placement of different adjuncts. Not all adverbials can occur in all positions (in all interpretations).

Both the flexibility and some types of possible restrictions can elegantly be captured if an optional adjunct-argument is placed in every place in the argument list where it can occur. This means that adjunct can be excluded from certain positions. Illustration of free adverb placement in (87), for example, is actually somewhat misleading. At least according to my intuitions *het boek voor Agnes* has to be one constituent in (87c). In (87b) it obviously is not, and in (87a) I suppose both analyses are possible. That means that the position of the adverb is less free than it appears to be at first sight. It is possible to distinguish different classes of adjuncts, which can occupy different positions, and their relative order can be constrained where they are adjacent. The number of verbal lexical entries remains the same. Each existing entry is enriched with a number of adjunct positions.

The semantics of adjuncts

Sentential adjuncts are semantically treated as operators. VP-adjuncts started out as operators too, but with the introduction of events in the semantics (see section 2.3.11 and chapter 2), we have experimented with letting them modify the event directly, making them very similar to arguments. Under the approach where the adjunct was the head and the VP the argument, a trick was needed, to make the event variable available for modification, because there is no lambda abstraction over it. Normally the semantics of an argument comes in as one variable, which is then put into the store. This means the semantics of the argument cannot be looked into. Therefore, a feature `eventvar` was added to the `synsem` path of all verbs, individuating the event variable.

For example, the verb semantics will be the following:

(88) *semantics from intransitive verb template*

```
{store:  { SemSubj binds C,
λBodyV.∃E.SemVerb(E) & EventType(E) & BodyV binds EV)}
body:  λTime.ThetaSubj(EV,C) & attime(EV, Time) }
```

Elsewhere in the template the path "`synsem:eventvar:EV`" will be present.

A manner adverb, for example, will then look in its argument for `synsem:eventvar:EV` and for the semantics, `sem:{Stores, λX.Body}`, rather than just `sem:SemArg`. Here it can take the different parts of the semantics as

separate variables, because the semantics will always have this form; stores, a lambda abstraction and a body. It is the form of the body that can not be predicted, because, for example, the number of theta roles can vary. And that is why we need to have the event variable. The semantics of the whole will then be: `{stores:Stores, body:λX.Body & atmanner(EV, Mod)}` The stores are copied from the verb and so is the variable that is abstracted over (i.e. the time). And to the body `atmanner(EV, Mod)` is added, where EV is the same variable that already occurs in the body as the variable for the event and Mod is the lexical meaning of the adverb itself.

If we switch to an approach where verbs look for adjuncts, we run into the problem that the verb needs to reserve slots in the semantics for these adjuncts. A rule is then needed that post-derivationally maps the adjuncts to the right slots. The assumption then is that for these types of adjuncts there is only one slot per event.

No such assumption is needed if we switch the primary and the secondary graph in the unification of a verb with an adjunct. Unification is guided by the grammar and normally the graph that is the head according to the grammar is made the primary graph in the unification process. The secondary graph unifies with a designated part of the primary graph. It is, however possible to do this the other way around, when the grammatically secondary category being consumed is an adjunct. The adjunct's graph will then be made primary in the unification, which means that the feature structures, including the semantics, can remain largely as they are. When there is no adjunct nothing happens, and when there is one it looks like the head in the resulting graph, even though the grammatical categories tell otherwise.

Summary

VP adverbials that modify the event can be treated as optional arguments. For NP adverbials this seems to be useful too. For sentential adjuncts (matrix, modal, attitude, ...) the situation is different. They will still be operators.

2.3.7 Coordination

The coordination algorithm is elaborately described by Cremers (1993). It is treated as an extra-grammatical procedure. The challenge in parsing and interpreting coordination is to determine where it starts and where it ends, especially when non-constituent coordination is to be covered, too. The algorithm still needs to be extended to also cover ellipsis. The present algorithm cannot efficiently handle more than one coordination per sentence. For covering cases of multiple coordination (and also multiple ellipsis) probably some heuristic strategies are needed.

Semantically a sentence containing one coordination is analyzed as two propositions. For example, (89a) is split up into (89b) and (89c) and then these two sentences are analyzed normally.

(89) a. Henk en Agnes werken.
Henk and Agnes work

'Henk and Agnes work.'

b. Henk werkt.
Henk works
'Agnes works.'

c. Agnes werkt.
Agnes works
'Henk works.'

Thus, a sentence of the general form (90a) comes out as (90b).

(90) a. X Y1 Coord Y2 Z
b. X Y1 Z (Coord) X Y2 Z

For coordination of subjects this works quite well. Even the option of a collective reading can be derived on the assumption that the two event descriptions can refer to the same event, which then has a complex agent.

In cases of coordination at the VP-level and lower, (90a) and (90b) are not always equivalent. Depending on the quantifiers involved, the entailment normally holds in only one direction. Cremers (1993) argues that the relation between (90a) and (90b) is never trivial. Non-monotone quantifiers (such as *exacly two*) then need to be analyzed as a conjunction of a monotone increasing and a monotone decreasing quantifier (*at least two and at most two*).

It would, however, be preferable to get the exact reading for the sentence, rather than something that either entails it or is entailed by it. This requires merging the two SLFs into one. In the resulting LF representations everything that scopes over the coordination occurs only once. Everything that scopes below the coordination occurs twice. Willis (2007) describes an implementation for NP-coordination in hole-semantics along these lines.

2.3.8 Anaphora

When the stores are applied, the binding of pronouns is also taken care of. The mechanism for the binding of pronouns and reflexives has been described by Visser (2005). She also proposes some changes in order to cover the difference between SE-anaphora (*zich*) and SELF-anaphora (*zichzelf*) (Reinhart and Reuland, 1993).

The binding algorithm is based on the classical binding theory from Chomsky's Government and Binding (Chomsky, 1981). For pronouns, the relevant principle is principle B, which states that pronouns must be free in their binding domain. Anaphora (reflexives) are subject to principle A, which requires them to be bound within their binding domain.

The algorithm works as follows. Pronouns are put in a special pronoun store. First an interpretation is derived in which all pronouns are free, since that is always grammatical. Then the possibilities for binding are investigated. A possible binder is a constituent of the category NP (equivalent to DP in modern versions of Generative Grammar) higher in the tree than the pronoun. The binder should not be a co-argument of the pronoun.

This reflects the notion of binding domain. A co-argument would be inside the binding domain and is therefore ruled out as a possible binder.

Reflexives are put in the reflexive store. Early in the apply_store algorithm, it is checked whether the reflexive can be bound, as sentences with unbound reflexives are ungrammatical. Also here, the binder must be an NP, but the positional requirements are of course different from those that apply to pronouns and their binders.

Both pronouns and reflexives must agree with their binders in person, number and gender.

Non-obligatory reflexives are treated as normal arguments. When the object of a verb is a non-obligatory reflexive the verb is interpreted as a relation between the subject and itself. There are two semantic arguments, but they both refer to the same entity.

Some Dutch verbs have obligatory reflexive arguments that do not seem to contribute to the meaning. For example, *zich schamen* 'to be ashamed' is taken to behave syntactically as a transitive verb, where the object is obligatorily reflexive, but semantically as an intransitive verb.

An interesting question is to what extent this approach is justified. Do these reflexives really not contribute to the semantics? It does not seem to be a total coincidence that some verbs have these reflexives and others do not. Cross-linguistically we see that verbs that are obligatorily reflexive in Dutch, tend to be so in, for example, Italian as well. Also the English counterparts tend to have a special form. They (for example) look like passives.

In addition, there exist alternations like the following.

(91) a. Het verveelt mij.
it bores me
'It bores me.'

b. Ik verveel me.
i bore me
'I'm bored.'

(92) a. Het ergert mij.
it annoys me
'It annoys me.'

b. Ik erger me eraan.
i annoy me there-on
'I'm annoyed by it.'

As yet, Delilah's analysis does not go above the sentence level. We will need to extend Delilah with a discourse component, in order to bind pronouns across sentence boundaries. Then also finding referents for other definite expressions becomes an issue. DRT may be a good candidate with which to handle anaphora, because it stays close to first order logic. In their textbook, Blackburn and Bos (2005, forthcoming) show how to build a system with DRT-semantics that is suitable for inference. Another option may be incorporating other existing systems. In general, Delilah's representations are rich enough to incorporate an existing algorithm for anaphora resolution

2.3.9 Extended Lexical Units

Extended lexical units (ELU's) are also called multi-word expressions or collocations. It is clear that there are many different types of ELU's. Sag et al. (2002) make a classification of multi word units that are problematic for NLP. Poß and van der Wouden (2005) propose a categorization of Extended Lexical Units (ELU's) on a cline, based on their flexibility. Some ELU's, are completely fixed. Examples are 'words with spaces', such as *by and large* and to a somewhat lesser extent idioms, such as *to kick the bucket*, which do enter the inflectional paradigm. Others are more flexible. Different ELU's can have varying degrees of flexibility. For completely fixed collocations a compositional analysis is not possible, but also not necessary. The more flexible a collocation is the harder it is to capture it in a fixed scheme with one fixed meaning. This also depends on the nature of the flexibility. Consider (93).'Here the word *bal* can be replaced by a whole range of other nouns (and every now and then, new variants appear), but that has no effect on the meaning.

(93) Hij snapt er geen bal van.
he understands there no ball of
'He doesn't understand a thing of it.'

In spite of the variation, it is possible to make one entry for all variants, in which the words that can occupy the noun position are listed. The semantics is kept non-compositional because the choice of the noun does not have any effect on the semantics of the whole unit[14].

A type of ELU, that has been implemented in Delilah in a systematic way, are verbs with a fixed PP-complement, such as *in stand houden* 'to preserve'. The meaning is attributed to the syntactic head. In most of the ELU's of this type that we implemented, this is a verb. The fixed argument comes with its own semantics, but this semantics does not contribute to the semantics of the whole. So, *houden* has the meaning *preserve* if it takes the fixed PP-complement *in stand*. The words *in* and *stand* are retrieved from the lexicon. Their templates are transformed to paths that are added to the specifications of the lemma, as shown in (94).

(94)
```
lemma( houden, verb, [trans_pp_fix],
       [id:Top+ID, head:concept:preserve,
       head:sem:preserve, head:phon:houden,
       synsem:extth:agent_of⌈Top+ID, A],
       arg(ID+ID1+1):sem:SemO,
       arg(ID+ID1+1):synsem:theta:theme_of,
       arg(ID+ID2+10):sem:SemS,
       sem:  { { [SemS*(ID+ID2)#A, SemO*(ID+ID1)#B,
             λBody.∃E.preserve(E) & event(E) & Body)
             *(Top+ID)#EV],[],[]},
             λTime.agent_of(EV,A) & theme_of(EV,B)
             & attime(EV, Time) }
```

[14]The choice of the noun does reflect a choice of register, but we do not consider this to be part of the kind of semantics we want to represent.

```
        |InStand],
        MorfForms ) :-
lemma(houden, verb, [trans_v, trans_v_sc], _,
      MorfForms),
lemma( in, pp, [pp_coll], SpecIn, _ ),
template( pp_coll, [id:ID+ID4|Template]),
construct( SpecIn, Template, TemplateIn ),
lemma( stand, noun, [abstract_noun], SpecStand, _ ),
template( abstract_noun, [id:ID4+ID5|TemplateH] ),
find( [synsem:cat:np], TemplateH ),
construct( SpecStand, TemplateH, TemplateStand ),
template_to_paths( TemplateStand, StandPaths ),
affix( arg( ID4+ID5+1), StandPaths, Stand ),
construct( Stand, TemplateIn, TemplateInStand ),
template_to_paths( TemplateInStand, InStandPaths ),
affix( arg( ID+ID4+2), InStandPaths, InStand ).
```

Many ELU's follow this pattern, with a bare noun. One problem is that *stand* as a bare noun does not occur outside this ELU (and a few others). Therefore, the generator should not freely use it as a normal mass/abstract noun. This can be done by giving it a different category, or by giving it a special feature that the generator is sensitive to, and that makes it be selected only if it is specifically asked for.

An ELU that is a bit more complex is *honger hebben* 'to be hungry' (lit. 'to have hunger'). *honger* is not the same kind of bare noun as *stand* above. It is a proper abstract mass noun[15] and can occur with a determiner, such as *geen* 'no', which has an impact on the semantics; *geen honger hebben* means 'not to be hungry'. The negation is on the determiner of the argument, but modifies the semantics of the whole construction.

This is not very surprising, since putting a negative determiner on the internal argument is a very usual way to negate an event in Dutch. For example, (95b) is the normal way of negating (95a), whereas (95c) is marked and only possible in particular contexts (e.g. for a contrastive effect).

(95) a. Bob eet een boterham.
Bob eats a sandwich
'Bob eats a sandwich.'

b. Bob eet geen boterham.
Bob eats no sandwich
'Bob does not eat a sandwich.'

c. (?) Bob eet niet een boterham.
Bob eats not a sandwich
'Bob eats not a sandwich.'

Employing the approach developed in section 2.3.4 for split scope effects gives us the right results for *geen* in ELU's, too. The prediction that the negation can then also move

[15] An abstract mass noun behaves like a mass noun, as long as it is unmodified. If it is modified by, for instance, an adjective, it can also occur with an indefinite article (*een*).

higher than just above the verb is born out. For (96) a reading is available in which the negation takes scope over the modal, the equivalent of a split scope reading.

(96) Bob kan geen honger hebben.
Bob can no hunger have

a. 'Bob can't be hungry.' $\neg > can$
b. 'Bob can be not hungry.' $can > \neg$

The problem, of course, is that *geen* is not selected for in the original entry of the ELU, so what we would need is an additional entry that includes *geen* and puts its meaning in the store.

The situation gets more complicated in the presence of an adjective such as *enorme* in *(een) enorme honger hebben* 'to be enormously hungry'. Many of the adjectives that occur in this ELU also occur as an adverb, resulting in the same meaning.

(97) a. Bob heeft (een) enorme honger.
Bob has (an) enormous hunger
'Bob is enormously hungry.'

b. Bob heeft enorm honger.
Bob has enourmously hunger
'Bob is enormously hungry.'

In (97b), the version with the adverb, it is unproblematic for the meaning of the adverb to modify the meaning of the ELU as a whole. For the version with the adjective, however, it is highly problematic to get such an interpretation under the present approach. For each adjective that the ELU can occur with, we would need a different entry for it. One lexical entry with one open slot for adjectives may also be an option, to cover at least a class of adjectives in one go, but that would still make it very difficult to get the semantics right, which would then have to be partly compositional.

An interesting point concerning all of these ELUs, is that they are listed in the lexicon under the verb, because that is the syntactic head that selects the rest. However, these verbs are in general light verbs and the central part of the meaning of the construction actually comes from the noun. In a traditional dictionary, such ELUs would be listed under the noun.

In the next chapter, I will propose to treat *honger hebben* not like an ELU, but like a light verb construction, where the core meaning is contributed by the noun *honger* and the light verb contributes the verbal properties, such as tense and aspect. This way the meaning of *honger hebben* can be built up compositionally in essentially the same way as *to be hungry* in English. An argument for this compositional approach is that the contribution of *hebben* ('have') is the same in e.g. *dorst hebben* 'to be thirsty' and the contribution of *honger* is the same in *honger krijgen* 'to get/become hungry'. In order to prevent overgeneration we just have to specify which light verbs go with which nouns.

An ELU with a lot of (semantically relevant) flexibility is the *way*-construction, described for Dutch by Verhagen (2003), and, from a computational perspective by

Poß (forthcoming). The construction is *zich een weg*[16] PP_{dir} $V_{intrans}$*-en*, equivalent to English *to* $V_{intrans}$ *one's way* PP_{dir}, and is illustrated by (98).

(98) Alice bluft zich een weg naar de top.
Alice bluffs herself a way to the top
'Alice bluffs her way to the top.

The meaning is roughly: to get to the place or point specified by the directional PP by V-ing. (Directional PP's will have to be analyzed as resultative.) This construction can be implemented with a fixed frame for the semantics in which the meaning of the verb and the PP have to be filled in, in specified places. The way it is implemented now, is that the event from the verb causes a moving event with the goal expressed by the PP. (A preliminary semantic representation for (98) is given in (99).) Thus the meaning of the whole is composed of the meanings of some of the parts plus an additional meaning part that comes from the construction.

(99) ιA.[top(A) & ∃B.[event(B) & move(B) & theme_of(B,alice) & goal_of(B,A) & ∃D.[bluff(D) & event(D) & agent_of(D,alice) & cause(D,A) & attime(D,E) & tense(D,pres)]]]

A problem is that the determiner of the *way*-DP is not entirely fixed. In particular, it can be *geen*. For the moment this is solved by putting the meaning of the determiner of this DP as a quantifier on the moving event.

A lot of work on ELUs still remains to be done, but I hope to have shown that we have some promising ways to compute the semantics of different types of ELUs. Like most lexicalist systems, Delilah can be seen as the embodiment of a construction grammar. The grammar is lexicon-driven, in the sense that all grammatically relevant specifications are stored in the lexicon. This makes the system particularly suitable to handle extended lexical units.

2.3.10 The representation of concepts

English words are used to represent concepts in the semantics, i.e. the lexical predicates in the logic of the representation. The underlying assumption is that we do not really know what words mean anyway. The concepts are just labels or placeholders. The reasons for using English words are mainly political. It stresses the distinction between concepts and word forms. (If Dutch words were used, this would appear less obvious.) Also, the use of English words makes the semantic output much more readable for an international audience. This is useful for presentations and demonstrations of the system.

However, representing concepts as English words may easily raise the suggestion of language independence, or of translation, which would be misleading. English words are not more neutral than Dutch ones, since English and Dutch are both natural languages. And translating Dutch to another language through semantic representations that make use of English words as concepts, would actually introduce extra errors, because of the

[16] *weg* can also be *pad* 'path', but this is semantically not relevant. In the present implementation the noun in this position is selected on the basis of its concept, not of its phonology.

additional translation step that is made. It is much better to try to map Dutch words with words of the target language directly.

For automated reasoning purposes, the most useful way of representing concepts, would be using the Dutch words themselves, or better, their citation forms. Next to containing syntactic information, representing a concept may be exactly what a word form does. Whatever information is available about the meaning of these words can then be looked up in sources such as dictionaries, encyclopedias, ontologies, and Wordnet.

Fortunately, word forms are in principle always available. Even if they do not appear in the semantic representation, they will occur elsewhere in the parse tree and are as such retrievable. The citation forms, do not occur in the parse tree, but a field dedicated to this information can easily be added. The concept field could also be used for this purpose.

Another advantage of using Dutch citation forms to refer to concepts and abandoning the use of English translations is that it improves the possibilities of (semi-)automatically extending the lexicon.

2.3.11 Event semantics

As will have become clear from the examples, the semantic representations in Delilah include reification of predicates in the form of event semantics. The format used is a variant of neo-Davidsonian event semantics based on the work of Parsons (1990).

One typical aspect of neo-Davidsonian semantics that we have already come across, are the argument roles that are part of the representation. In the older semantic format of Delilah, theta roles have always been present in the parse tree and they were referred to by certain lexical rules, such as the construction of passive forms. Since the introduction of event semantics in Delilah, theta roles have become part of the semantic representation too. Roles used are: agent_of, source_of, location_of, experiencer_of, goal_of, theme_of, patiens_of, instrument_of, place_of, manner_of, and time_of. (Another value that occurs in this field is 'expletive', but that one does not appear in the semantics. It actually indicates that the argument in question does not have a theta role.) A weakness of the way these roles are used in Delilah is that it is not very systematic. There appears to be no clear policy as to what criteria there are to determine what role should be used. Roles for the arguments are chosen rather intuitively and per lemma. I will come back to this point in the next chapter.

As we have seen in the example, tense introduces an operator, such as 'AtPres' or 'AtPast' that binds a pronoun, which is used in an 'at-time' relation to the event. The idea to treat tense anaphorically originates from Hinrichs (1986). The temporal pronoun in e.g. infinitival complements is bound by the tense operator of the matrix verb. For the rest temporal relations are not very much worked out.

The introduction of event semantics has made the representations flatter, splitting up the meaning of a sentence into a collection of conjoined 'small clauses'. Event semantics has been implemented not only for verbs, but also for nominalizations. In addition the approach has been extended to intersective predicative adjectives and their corresponding abstract nouns. Both the principle of 'conjunctivism' and its extension beyond verbs are of benefit for deriving wanted entailments. The details of the implementation of event semantics in Delilah form the topic of the next chapter.

2.3.12 Disambiguation

In section 2.3.5, I have discussed scope ambiguity, its underspecification and its prospects for disambiguation. The present section is about other types of ambiguity. Delilah thus far does not have a statistical component. This means that it is designed to compute all possible analyses of a sentence that the grammar allows, but does not have statistical means to decide which is the most likely analysis. A recent development is that it can select the least complex SLF as the best analysis. This favors argument readings over adjunct readings (where both are possible), because the semantics of an argument unifies with an element already present in the store, whereas an adjunct adds a new element. It also favors idiomatic readings over literal ones. However, this still leaves many ambiguities unresolved, for example ambiguities in PP attachment, between different binding options for a pronoun or between canonical order and inverted readings (object fronting, resulting in OVS order). Note that a statistical component would only help resolving syntactic ambiguities. There are no corpora on which a statistical algorithm could be trained to determine what is the most likely semantic analysis, although there are some that target very specific isolated aspects of semantics, such as role-labeling or anaphora resolution. It is probably also early to develop a corpus that is annotated with full semantic representations, because there is still a lot of room for improvement of such representations. This means that at least for favoring collocational readings over literal readings there is probably no syntax-statistics based alternative. See section 2.3.9 for ELUs in Delilah.

2.4 Robustness

Delilah is being made more robust in several ways. This is necessary if we want the parser to parse whole texts. The lexicon and the grammar will need to be extended, but even then, they will not cover everything. Therefore, strategies are needed to deal with unknown words and grammatical constructions. If all else fails, a partial parse can be given. Whereas Delilah was originally designed to only recognize sentences, it turned out to be relatively easy to let the system give partial parses as well. If it is unable to parse the whole sentence, it tries to construct phrases that contain the largest possible strings from the input.

2.4.1 Extending the lexicon and the grammar

Delilah started out with a toy-lexicon, to which every now an then words were added, all manually, so for a long time the lexicon was very small. Possibilities of using external information sources to extend the lexicon are currently being explored. One such source is the lexicon of the Alpino parser. Alpino has a big lexicon that covers a newspaper corpus. Alpino entries can be partially automatically transformed into Delilah entries. The tags that are used in Alpino can be mapped onto templates in Delilah. The Alpino lexicon, however, does not contain any semantic information, as Alpino is only a syntactic parser. Missing information includes theta roles, event type and semantic

concept. For the theta roles defaults can be used, e.g. per template. The event type can either be left unspecified, or a default can be chosen (possibly also per template). In order to stick to the English-words-for-concepts policy, one would need to automatically use a machine readable Dutch-English dictionary. It seems very hard to do this. One of the problems is, for example, how to choose between different options. Of course one could try to make a different lemma for each translation equivalent, but this would create a lot of ambiguity, and if the meanings are close to each other, this may not be necessary. Alternatively, one could also use the Dutch word form.[17]

The grammar is also extendable. A number of verbal templates have, for example, been added in order to accommodate all the verb classes from Alpino. New modes can be added as well. The only principled limitation to the number of modes is that it should be finite.

2.4.2 Dealing with unknown words

In the past, Delilah would stop and give up on the sentence, as soon as it encountered a word that was not in the lexicon. Now that it is also possible to give partial parses, it of course makes sense to just skip unknown words, or assume by default, that they are NPs.

In principle it should also be possible to guess the category of an unknown word, especially if there is only one unknown word in the sentence. The categorial grammar should certainly be able to decide which category is still needed in order to form a sentence together with the other words. Such an algorithm would be related to the coordination algorithm, which is described in Cremers (1993). Default semantic structures can be used for each category.

A similar strategy can possibly be applied to gaps in the grammar. If a certain configuration is not covered by the grammar, it may help to try to consume one of the arguments under a different mode than specified. For this, Delilah would need to be able to diagnose that a parse crashes on one particular mode. If an argument that was believed to be obligatory is not found, Delilah may want to assume that the head can also occur without this argument. Or if the arguments appear in a different order than expected, this order can be added.

2.5 Other computational semantics systems

Delilah is so far the only system that gives semantic representations for Dutch sentences. There are some syntactic parsers for Dutch, such as Alpino (Bouma et al., 2001), developed in Groningen, and Amazon (Coppen, 1995), developed in Nijmegen.

For English there are a few systems that provide (deep) semantic representations. An influential one is English Resource Grammar (ERG) by the LinGO project (Copestake and Flickinger, 2000), an HPSG parser that makes use of minimal recursion semantics (MRS)(Copestake et al., 2005). MRS is an underspecified representation formalism,

[17] Another strategy that was considered, was to assign vague concepts, such as *thing* for nouns and *action* for verbs. This, however would give rise to a lot of unwanted entailments. This may be prevented by numbering them, for instance.

which includes constraints on scopal relations. It was developed with machine translation in mind and therefore uses somewhat flattened structures. The difference between an underspecified representation and a fully specified one is only in the constraints. As constraints are added, fewer readings become possible, till there is only one left[18]. For automated reasoning it needs an adapted mechanism for reasoning with underspecified representations, since the output is not in predicate logic, but in this particular MRS formalism.

Verbmobil (Wahlster, 2000), a speech-to-speech translation system for German, English and Japanese makes use of the LinGO ERG parser. The HPSG parser is applied in combination with a statistical parser and a chunker, where the chunk parser produces the most robust and the HPSG parser the most accurate results. The semantic formalism used is LUD, a description language for underspecified discourse representations (Bos et al., 1996). The Verbmobil system is the result of a huge project with many collaborating parties. It is very elaborate comprising a wide range of techniques, from prosodic to discourse analysis.

XLE (Maxwell and Kaplan, 1993) is the parsing project by Xerox PARC. It is based on Lexical functional grammar (LFG) (Dalrymple, 2001), and the semantic formalism is glue semantics. The XLE parser is also used by Powerset for their internet search engine.

Bos et al. (2004) describe a CCG parser that parses part of speech tagged text. This way it is very robust and barely needs a lexicon. It is only for closed class lexical items that the lexical semantics is spelled out for each lemma individually. Open class lexical items are assigned a lambda expression based on their CCG category and the lemmatized wordform. Davidsonian event semantics is used. Their paper does not explain from where they get the theta roles that are used in the representation. Semantic representations are in first order logic and in a later version in a first order logic translation of DRT. Any information that one would like to add about the semantics of lexical items, which goes beyond the standard semantic structure that is assigned on the basis of the POS-tag, should come from some other source, for example a collection of meaning postulates. In Bos (2005) world knowledge from Wordnet is used to support inferencing. One of the current research efforts is to make the semantics more detailed. A possible limitation of this approach is that it is dependent on the finegrainedness of POS-tags. Adaptations to the tag-set are not easily made, because the tagger needs to be trained on an annotated corpus. If you want to make additional distinctions that are useful to your grammar and semantics, you need to re-annotate the corpus for these categories and re-train the tagger. Delilah, in contrast, is strongly lexicalist. However, if techniques are developed to automatically upgrade Delilah's lexicon, the difference might become less important.

Mooney (2007) reviews several systems he and his group developed for learning semantic parsers from corpora annotated with formal meaning representations.

In general, interest in deep semantic analysis is increasing.

[18] In principle it is probably possible to combine constraints in such a way that no reading is possible anymore.

2.6 Summary and conclusion

Delilah is a semantic parser and generator based on Combinatory Categorial Grammar with modes of composition. Its lexicon consists of typed feature structures. Delilah is at present the only deep semantic parser for Dutch. The combination of categorial grammar and feature structures results in a strong lexicalist approach. Delilah has a huge lexicon, a large portion of which is produced by morphological rules. The semantic component applies the composition of lambda terms to yield first order logic formulas.

A storage mechanism based on the work of Cooper and Keller is used to compute the different semantic readings of a sentence. Binding of pronouns is covered within the sentence. SLF is also underspecified for that. The device of storage is an effective tool for allowing one syntactic structure to have more than one reading. For partial disambiguation and reasoning with underspecified representations it is less effective. The flat semantic format that will be discussed in chapter three offers a different way of underspecification, or compact notation of different readings. No ideal method exists either for underspecification, or for disambiguation of scope.

I argued that for correct analysis of split scope constructions, the negation involved needs to be stored as an operator. Other operators do not need to be stored, because their scope is fixed by syntax. The algorithm that computes the different scope options of quantifiers, does need to be sensitive to such operators. I proposed an algorithm that prevents generating the same reading twice.

There are several problems with adjuncts, both in the syntactic and the semantic domain. I discussed them and proposed some tentative solutions. VP-adjuncts need to be treated as optional arguments syntactically. Semantically the adjunct acts as the head, because its graph gets the role normally assigned to the head category's graph in the unification process.

Delilah is, as a construction grammar, able to handle extended lexical units. I discussed the implementation of several types of these. Their idiomatic readings can be distinguished from the literal alternatives, by their less complex SLFs.

I also reported on efforts to make the parser more robust. Extensions of lexicon and grammar are being planned. Partial parses have been made possible. A way to guess the category of unknown words has been conceptualized.

I have pointed out that using Dutch words for concepts has advantages over using English ones.

More work is needed in all of the areas mentioned. Most of the issues are complex and important enough to have a separate thesis devoted entirely to it. Instead, my intention has been to give an overview of the current status of the parser.

In the next chapter, I will discuss event semantics, which was briefly introduced in section 2.3.11.

Chapter 3

Events in the semantics

This chapter discusses event semantics in Delilah. Events were introduced to get flatter structures with more handles for inference. The use of event semantics, usually neo-Davidsonian is by now common in computational semantics. The semantic role labeling, typical for shallower approaches, is a related phenomenon. The identification of events and their participants offers valuable information which one wants to be able extract from text. In event semantics, the event argument reifies predicates and thereby makes them addressable and quantifiable. This has advantages for reasoning with modifiers. The neo-Davidsonian approach has the additional advantage that it avoids complications to do with the arity of predicates. All together event-semantics makes several important inference patterns to follow naturally, which otherwise would need the help of meaning postulates. For these reasons it was felt that an implementation of event semantics was indispensable for Delilah.

Also on theory side, event semantics is an important field of research nowadays. In particular, explanations for the aspectual behavior of verbs are sought in properties of the event and its internal structure. Our discussion here barely scratches the surface of this elaborate field. The theoretical work generally targets a very limited subdomain for a very detailed analysis. In our implementation, first a basic across the board implementation was needed. From there on in future one can try to put in more state-of-the-art theory for specific groups of verbs or specific phenomena.

The implementation started out from the verb classes we have in Delilah, based on the templates. The template is the natural place to introduce a frame for the event semantics, containing the event, the quantification over it and slots for the participant roles.

The basic case are verbs that introduce an event with one, two or three NP or PP arguments that get roles and without further complications. There are also cases where the complement of the verb is a VP or an embedded clause or small clause, containing itself an event (or state), so one has to consider what the relation should be between the two complex structures. Often intensionality plays a role in such cases. Also, it can happen that there is evidence for the event structure introduced by a verb to be more complex, or that an event is introduced by a word that is not a verb. I will discuss per case what choices have been made.

Section 1 provides some background on Davidsonian and neo-Davidsonian event representation and discusses problematic issues of a general nature. In section 2, I discuss the implementation of neo-Davidsonian event semantics for verbs in Delilah, starting out

with the most basic case and then discussing several verb classes that need more attention. Section 3 shows how event semantics is implemented for nominalizations. The challenge here is the interpretation of the participants which can be expressed in many ways. In section 4, I discuss underlying states and show how adjectives and their corresponding abstract nouns can be analyzed parallel to verbs and their nominalizations, but with states instead of events. Finally, I show how the combination of the type of implementation that I used for support verbs of nominalizations with an underlying state approach suggests a compositional analysis of light verb constructions, which used to be implemented as collocations.

I make the distinction between events and states, mainly because the latter are more controversial, so I want to argue for them separately. Events and states are both eventualities. The term *events* is sometimes also used to refer to the whole class of eventualities.

The implementation of event semantics prepares the ground for the flat logical form discussed in the next chapter. It is a first step towards flatter semantic representations.

3.1 Neo-Davidsonian event semantics

Davidson (1967) proposed that simple event sentences should be analyzed as asserting the existence of an event of a type specified by the semantics of the sentence. He used the idea of Reichenbach (1947) that action sentences have an existential quantifier binding the action-variable. Events are introduced as entities about which an indefinite number of things can be said. He takes predicates of action verbs as containing an extra place for the event variable. Thus the logical form of (100) will be saying something like 'there is an event x such that x is a flying of my spaceship by me and x is to the Morning Star'. Advantages of this notation are that (100) and (101) together entail[1] (102) and, most importantly, that (100) entails (103).

(100) I flew my spaceship to the Morning Star.
($\exists$x) (Flew(I, my spaceship, x) & To(the Morning Star, x))

(101) the Morning Star = the Evening Star

(102) I flew my spaceship to the Evening Star.
($\exists$x)(Flew(I, my spaceship, x) & To(the Evening Star, x))

(103) I flew may spaceship.
($\exists$x)(Flew(I, my spaceship, x))

The advantages become even clearer in sentences with more modifiers. (104a) entails (104b) through (104f), but (104c), (104d) and (104e) together do not entail (104a), because they may each refer to different events.

(104) a. Jones buttered the toast in the bathroom at midnight with a knife.

[1] A entails B iff whenever A is true, B is necessarily also true.

b. Jones buttered the toast.

c. Jones buttered the toast in the bathroom.

d. Jones buttered the toast at midnight.

e. Jones buttered the toast with a knife.

f. Jones buttered the toast in the bathroom at midnight.

Davidson's events are quite widely accepted. In different semantic subfields it is common to assume underlying events. Also some more syntactically oriented work makes reference to events. Relatively many computational semantic formalisms incorporate some form of event semantics. Nevertheless, opinions on the details of the representation differ. Davidson's original proposal consisted of the addition of an event argument to action verbs. Neo-Davidsonians have proposed an eventuality argument for states as well. Also, they started using thematic role relations in the representation to link the traditional arguments to the event. Within the neo-Davidsonian stream there are different ways to link the verbal concept to the event variable. A potential issue of debate that applies to both variants is the status of the existential quantification, usually assumed to bind the event variable. In this section I will discuss these issues, except for that of underlying states, which will be elaborately discussed in section 3.4.

3.1.1 Naming the event

An event is usually represented as, for example, *work(e)* for a work event, where *e* is a variable bound by a quantifier and *work* is a name for the verbal concept. That is, events are named in the same way as entities are (cf. *book(x)*). The event name is a predicate over a variable. Instead of 'work', also 'working' occurs in some approaches. I think that, in principle, the difference is not interesting, as concept names are arbitrary. For reasons of inference, it is of course important to be consistent, that is, to use the same concept for all word forms.

In Delilah, there are no dedicated variables for events. Therefore an extra predicate was added, yielding *event(x) & work(x)*. The information that something is an event appears to be relevant, because we can infer that something happened if there was an event at the relevant time (and place). Introducing this extra predicate prompts the question whether these two predicates over the event variable are actually of the same kind and should be represented in the same way.

Jurafsky and Martin (2000) use the is_a relation to relate the event variable to the verbal concept: *is_a(e, work)*. This inspired the use of *concept(e, work)* in Reckman and Cremers (2006). If it is extended to (other) entities as well (*concept(x, book*)), it may have the advantage that there are only functional predicates and that concepts that may need to be looked up in, e.g. , an ontology are in a predictable place in the representation.

The option *event(e, work)* draws a parallel with the representation of thematic roles in a neo-Davidsonian approach. It seems convenient because it packages the two pieces of information together. Nevertheless, it is somewhat unintuitive because if we read it the same way as with the thematic roles, then it says that the event of *e* is *work*. But what is *e* then?

I take it that *work* is a concept and *event* is a superconcept, a hypernym. It provides extra ontological information. This is consistent with the present representation where both pieces of information are of the same kind; both are conceptual information. It is, however, a more principled approach, to make this extra information available in an external ontology, rather than include it in the semantic representation of a sentence. On the other hand, if it is information that is very often needed, it may prove more efficient to standardly include it in the representation, instead of having to look it up. In that case, it may acutally be usefull to use a notation that is more common in artificial intelligence, where an event would be represented in the format of *event(e, work)*, but an object also in the same format; *object(x, book)*. This has the advantage that if this appraoch is taken consistently, all predicates in the representation will be functional, drawn from a limited set, and all concepts will appear in argument positions. This kind of normalization makes it easier to decide which elements in the representation are open class concepts which may need to be looked up in an ontology or encyclopedia. This will need to be evaluated when experimenting with inference and ontology use. The present choice was made in order to let our representations include the same information as the representations proposed by e.g. Parsons (1990) do. The information in the semantics is not used for purposes of argument selection. For this, a syntactic feature is included that conveys the event status of an argument.

3.1.2 Existential closure of events

The event argument is normally represented as bound by an existential quantifier, which is generally assumed to be the result of existential closure. I have found no alternative proposals in the literature. The use of choice functions is a more general alternative way to deal with indefinites, which probably can also be applied to events.

Amsili and Hathout (1996), building on Kamp and Reyle (1993) and Asher (1993), convincingly argue that the existential quantifier over the event scopes below sentential negation and that negated events are not states. That is, (105) means that there was no event of Alice going to Paris at the relevant time, and not that there was an event (or state) of Alice not going to Paris.

(105) Alice didn't go to Paris.

Of course, this is a simplification. When focus is taken into account (cf. Herburger 2000), (105) could also mean any of the following, depending on what is in focus.

(106) a. Alice did something, but it wasn't going to Paris.
b. Alice went somewhere, but it wasn't Paris.
c. Someone went to Paris, but it wasn't Alice.

These readings are all more specific than the one proposed above. All of these entail that there was no event of Alice going to Paris. This is therefore the most general reading and the safest one to use in absence of focus information. It does not entail that someone went somewhere, which all the others do. When it becomes clear in the context that

one of the other readings is intended, for example (106b) in (107), we still can entail all the information needed, while keeping the general reading, because the disambiguating sentence actually disambiguates by providing the additional information.

(107) Alice didn't go to Paris. She went to Berlin.

The following examples by Dölling (2005), however, suggest that there can be states that are characterized by something not happening.

(108) a. Het landschap bleef niet onveranderd.
the landscape remained not unchanged
'The landscape did not remain unchanged.'

b. De dijk weerstond de waterdruk niet.
the dike withstood the water pressure not
'The dike did not withstand the water pressure.'

In the implementation in Delilah, the event is introduced, with the existential quantifier, in the store of the verb semantics, as an argument. This is illustrated in the general scheme in (109). The stores of the event argument are empty, and are omitted here. Information that is only about the event is included in the stored quantifier. Thematic role relations involve a variable that is bound by another quantifier. Therefore they are introduced in the body.

(109) store:{λR.$\exists$E. event(E) & ... & R *binds* A,
Arg_1 *binds* B, ..., Arg_n *binds* K}
body: λT. role_1(A,B) & ... & role_n(A,K) & attime(A,T) ...

Introducing the event in the store means that it can exhibit scope ambiguities with respect to its arguments. There is some evidence that this is indeed the case. For (110a) the readings (110b) to (110e) seem to be distinguishable. (110c) is a more general version of (110b) and (110e) is a more general version of (110d). That is, (110b) entails (110c), because if there is a event of carrying two tables, then for each of the tables there is an event of being carried, even though it is in both cases the same event. Thus, (110c) is the most neutral surface scope reading and (110e) is the most neutral inverse scope reading. There also is a scenario that makes both (110b) and (110e) true, as well as a scenario that makes both (110c) and (110e) true.

(110) a. four girls carried two tables

b. $4 > \exists > 2$
for each of the four girls there was an event of her carrying two tables (4 girls, 4 events, 2 to 8 tables)

c. $4 > 2 > \exists$
for each of the four girls there were two events of her carrying a different table (4 girls, 8 events, 2 to 8 tables)

d. $2 > \exists > 4$
for each of the two tables there was an event of four girls carrying it (2 tables, 2 events, 4 to 8 girls)

e. $2 > 4 > \exists$
for each of the two tables there were four events of a different girl carrying it (2 tables, 4 to 8 girls, 4 to 8 events)

Readings where the event has widest scope only seem to allow for collective readings, where there are no more than 4 girls and 2 tables involved. (For further discussion of this type of data, see e.g. Verkuyl and van der Does (1991).)

Also, for verbs that only have a subject and no other participants, it is not very clear whether there is a (meaningful) scope ambiguity, for example in (111). Some well-known predicates, such as *meet* and *gather* only allow for collective readings.

(111) All women worked.
for every woman there was a (possibly distinct) working event
there was one single event of all women working

Because of the way the lexicon is built, all derived forms of a verb also contain the event argument in their semantics. This is for example the case for adjectival forms. Here however, the event quantifier cannot raise out of the complex NP (e.g. (112a), just like the indefinite in (112b).

(112) a. alle gekochte boeken
all bought books

b. alle boeken met een foto op de voorkant
all books with a photo on the front(cover)

Lexical nominalizations are not derived from verbal entries because their form and to some extent also their meaning are not predictable on the basis of the verb. Nominalizations and their events are discussed in section 3.3.

What exactly are the restrictions on the scope of the existential quantification over the event needs to be sorted out more in detail. The important thing is that when it is in the store, it can scope like any other quantifier and restrictions can be formulated as needed. The present strategy is to keep the event low, as this yields the most general readings.

Treating the event as definite rather than existential would make it scope insensitive. This creates a problem under negation. The negation of such an event would then entail that there is no unique event with the relevant properties. This leaves room for the existence of some events that are not unique and have the relevant properties.

3.1.3 Thematic roles

With the introduction of event semantics in Delilah, thematic roles started to be used in the semantic representations, that is, we used a neo-Davidsonian version of event semantics, mainly based on Parsons (1990). This is, however not the only possible way to implement event semantics. The main problem of semantic roles, is that it turns out to be very difficult to assign them consistently. This was of course always a problem, but it gets more important now that the thematic roles are used in the semantic representation, because in that way they become relevant for inference. In this section, I discuss whether thematic roles should be used in the semantics of Delilah and if so, which set of roles should be used, and how it can be determined which participant should get which role.

Davidsonian versus neo-Davidsonian representations

I will first briefly motivate the use of thematic roles in event semantics, before going into theories of thematic roles. In Davidson's original proposal, an event argument was added to a traditional predicate, making an $n + 1$ place predicate out of an n place one, illustrated in (113b). Others, such as Parsons (1990), represent arguments by means of thematic relations to the event, which is more similar to the representation of adverbials, resulting in a higher degree of decomposition, illustrated in 113c.

(113) a. Brutus stabbed Ceasar with a knife.
b. stab(brutus, ceasar, e) & with(e, a knife)
c. ∃e [stab(e) & agent(e, brutus) & theme(e, ceasar) & with/instrument(e, a knife)]

As Herburger (2000) points out, the latter approach is advantageous in cases where there are theta-marked arguments that are genuinely optional. Meant by 'optional' is that they do not have to be lexically realized and are not semantically implicit when they are absent. For example, (114a) does not entail that Brutus stabbed someone, (114b) does not entail that Alice wrote a note to someone, and (114c) does not entail that he said something to someone.

(114) a. Brutus stabbed and missed.
b. Alice wrote a note.
c. He said something.

A classical Davidsonian representation would require a two place version of *stab* for (114a) next to the three place version above. But then (115a) would not entail (115b). This is because there are no entailment relations between predicates with different arity: pred(x,y,z) and pred(x,y) are two different and unrelated predicates that accidentally have the same name.

(115) a. Brutus stabbed Ceasar.
b. Brutus stabbed.

Similarly for (116a) and (116b).

(116) a. Alice wrote a note to Bob.
b. Alice wrote a note.

A second argument for separating the arguments in this way is put forward by Schein (2002). He argues that the most natural reading of a sentence like (117) is "There was a teaching in which three video games were the teachers, and that teaching resulted in every quarterback's learning two new plays each." This is an argument for having a separate clause at least for the agent role.

(117) Three video games taught every quarterback two new plays.

Kratzer (forthcoming) shows that this "group reading argument" only holds for agents and not for themes[2]. The optionality argument above, however, does hold for themes (e.g. (114a)).

A third argument is that we may want to infer that the agent of an event did something. Both (116a) and (116b) entail (118).

(118) Alice did something.

This follows if the representation of (118) is (119).

(119) $\exists$e. agent(e, alice)

Here also the content of the role matters.

An additional advantage of the representation with semantic roles is that arguments and adjuncts are represented in essentially the same way. This way, for example, prepositional phrases can be represented in a consistent way, no matter whether they are selected and obligatory or not selected or not obligatory.

To summarize, the use of roles avoids variation in arity/valency of predicates, which is good if there are optional arguments, separable agents seem to be needed for group readings and thematic roles help in deriving, for example, that the agent "did something". Therefore, I conclude that inferencing can benefit from neo-Davidsonian event semantics, i.e. with thematic roles.

Approaches to thematic roles

As it was judged to be beneficial to use thematic roles in the semantic representation, we will need to decide what set of roles to use, and what criteria to apply for deciding which participant gets which role. There exists a wide variety of approaches to thematic roles. This is at least partly because the study of thematic roles can have different purposes. A quite common purpose in syntactic research is to explain matters of argument selection, also known as the linking problem: Why are which participants expressed how in the syntax? Another purpose is to gain insight in conceptual structure (Jackendoff, 1983, 1987). This is common for the more cognitive approaches. Thematic roles are also used to pursue criteria for event identity. On the assumption that in an event there is exactly one filler for each role (bi-uniqueness), events cannot be identical if they have different fillers for a particular role (e.g. different agents) (Carlson, 2001). My main concern will be entailment relations between events.

Views on thematic roles vary from those that posit a small number of general roles, such as agent and patient, to those that let each verb assign its own verb-specific thematic roles.

Letting each verb assign its own roles reflects the idea that in every kind of event the exact nature of the involvement of the participants is different. Related to this is the point of view that thematic roles also include information about what are suitable fillers (Altmann, 1999). Verb-specific roles, of course, do not allow any kind of generalization, unless they are built up of more general features, or are organized in role types.

[2] See Cremers (2001) for a list of other observed restrictions on collective readings

Another disadvantage[3] of verb specific roles is that they do not automatically allow inferences from more specific to less specific verbs (in hyponym-hypernym relation). If *run* and *go* both assign the same role to their subject and to their directional argument, then you only need to know that running is a specific kind of going in order to conclude that (120a) entails (120b).

(120) a. Bob ran to the store.
b. Bob went to the store.

Information on hyponymy relations between verbs is typically available in resources like WordNet. If the two verbs each assign their own roles to their participants, a mapping of these would be needed in addition (e.g. a runner is a specific kind of goer). The latter kind of information is less readily available. An interesting case is FrameNet (Baker et al., 1998), which has rather specialized roles, but does have a mapping between specific and more general frames.

A disadvantage of role types, on the other hand, at least in the way that they are sometimes applied by researchers working on the linking problem, is that roles are taken to change in alternations. The typical case are spray/load alternations. It has been argued that in (121a) *the truck* is the theme, whereas in (121b) *hay* is the theme and *the truck* is the goal. (Note that it is the pattern that is of interest here, not the labels that are chosen for the roles.)

(121) a. Alice loaded the truck with hay.
b. Alice loaded hay onto the truck.

For inference, this is a particularly bad solution, since (121a) entails (121b). The same goes for (122a) and (122b). When they are analyzed as (Schein, 2002) does, with *music* being the theme in (122a) and *the clarinet* being the theme in (122b) the entailment is lost.

(122) a. Ray played music on the clarinet.
$\exists$e. play(e) & agent(e, ray) & theme(e, music) & on(e, the clarinet)
b. Ray played the clarinet.
$\exists$e. play(e) & agent(e, ray) & theme(e, the clarinet)

Note that intermediate positions between verb specific roles and a very limited number of generalized roles are harder to realize with a Davidsonian than with a neo-Davidsonian approach. In a system without explicit roles, the number of roles is either limited to the number of places of a predicate, i.e. the first place role, the second place role, etc., or completely verb specific (the first place role of verb A is a different one than the first place role of verb B). In the latter case, a mapping may be specified, e.g. the first place role of verb A corresponds to the second place role of verb B. In the former case, the possible assignment of roles is more restricted than in a similar neo-Davidsonian approach, because a two-place predicate cannot assign a third place role, for instance.

[3]This is of course only a disadvantage if we agree that such entailments between different verbs actually hold.

Dowty's proto agent and proto patient properties

A very influential position is that of Dowty (1991), who focusses on the problem of argument selection. He introduces two proto-roles: proto-agent and proto-patient. Depending on how many proto-agent and proto-patient properties a participant has (also in comparison to other participants) it is mapped to a certain syntactic position (subject, object, oblique). The participant with the most agent properties becomes the subject and the participant with the most patient properties becomes the direct object. The properties are things that are entailed for a participant. The Proto-Agent and Proto-Patient properties are listed in (123) and (124), respectively.

(123) Contributing properties for the Agent Proto-Role:
   a. volitional involvement in the event or state
   b. sentience (and/or perception)
   c. causing an event or change of state in another participant
   d. movement (relative to the position of another participant)
   e. (exists independently of the event named by the verb)

(124) Contributing properties for the Patient Proto-Role:
   a. undergoes change of state
   b. incremental theme
   c. causally affected by another participant
   d. stationary relative to the movement of another participant
   e. (does not exist independently of the event, or not at all)

Dowty's theory elegantly explains several alternations with partially symmetrical predicates, such as *kiss* and *collide*. For *kiss*, he argues that in (125a), where both Bob and Alice are in subject position, volitional involvement in the event is entailed for both of them, and hence the event is symmetrical, both participants are involved in exactly the same way. In (125b), Bob appears in subject position and Alice in object position, with the result that volition is only entailed for Bob. This accounts for (125a) entailing (125b) and (125c) but not vice versa. ((125b) and (125c) together do not entail (125a) either, because they may refer to different events.)

(125) a. Alice and Bob kissed.
   b. Alice kissed Bob.
   c. Bob kissed Alice.

The same goes for *collide*, except that here the additional Proto-Agent property entailed for the subject is that of movement. This is shown by the oddness of (126c). It is odd because it entails that the wall moved. (126d) on the other hand, does not have such an entailment. Since the oblique participant does not have any properties that the subject doesn't have, it follows that (126a) entails (126b) (and (126c) entails (126d).

(126) a. The car and the truck collided.

b. The car collided with the truck.

c. (#)The car and the wall collided.

d. The car collided with the wall.

Note that some verbs that display this kind of pattern are fully symmetrical (at least in truth-conditional terms). For example, (127a) entails (127b) and (127c) and (127b) and (127c) each entail both each other and (127a). This means that the transitive version of *marry* must assign exactly the same thematic properties to its subject and its object.

(127) a. Alice and Bob married.

b. Alice married Bob.

c. Bob married Alice.

That (127a), (127b) and (127c) are truth-conditionally equivalent, of course does not mean that they are identical. With respect to information structure they are certainly different. Dowty, however, argues against information structure being reflected in thematic roles. I agree that this kind of information must be kept separate. If, for instance we want to know whether Bob ever married anyone and we find sentence (127b), we want to be able to conclude that Bob married Alice, and we do not care that this sentence may have been intended to tell us something about Alice, rather than about Bob.

For psych verbs, there is a draw, because sentience and causation are both proto-agent properties and the participants do not differ in other properties. Then it can go either way and may differ per verb; in (128a) the subject is sentient and the object causing, while in (128b) they have changed place.

(128) a. Alice fears the monster.

b. The monster frightens Alice.

In the spray/load alternation, according to Dowty, the participant in the object position has the property of incremental theme, in addition to its other properties, that remain the same in both configurations. So, in (129a) it is *the truck* that has the additional property of incremental theme, and in (129b) it is *the hay*. This correctly predicts that neither (129a) entails (129b), nor the other way around. In (129a) the truck ends up full, while some of the hay may be left over. (The option of hay being left over is pragmatically a bit odd. It helps to imagine that the purpose of hay is to fill trucks. In the spray equivalent (130a) it is much clearer that not all the paint needs to be used.) In (129b) the truck does not need to end up full, but all of the hay referred to ends up on the truck.

(129) a. Alice loaded the truck with the hay.

b. Alice loaded the hay onto the truck.

(130) a. Alice sprayed the wall with the paint.

b. Alice sprayed the paint on the wall.

Summarizing, the advantages of Dowty's theory are that not all agents and all patients are the same, that thematic hierarchy effects follow and that there can be a draw (e.g. psych verbs).

What Dowty's proposal does not explain, however, is that (131a) as well as (129a) entail (131b) (though not the other way around). In addition, it is not clear how to derive that (132a) entails (132b).

(131) a. Alice loaded the truck with hay.
b. Alice loaded hay onto the truck.

(132) a. Alice loaded hay onto the truck.
b. Alice loaded the truck at least partially with hay.

The incremental theme is defined as an NP that can determine the aspect of the sentence, since the parts of the event correspond to parts of the NP referent that are affected by the action. To correctly represent aspectual effects, more decomposition is needed than just thematic roles. It is necessary to assume, for example, a resulting subevent. Dowty's properties may be precise enough to explain argument selection, but they are not precise enough to explain all entailment patterns.

Below are some inferences that we would like to have for *Alice loaded the truck with Q hay*.

(133) inferences of: *Alice loaded the truck with Q hay.*
a. Alice moved some hay. (*some hay* being part of what is referred to by *Q hay*, possibly all of it)
b. Some hay ended up on the truck.
c. The truck ended up full of hay.

This suggest that a representation is needed that roughly contains the following information.

(134) Alice moved part (or all) of Q hay (but non-zero amount) with the result that it ended up on the truck and that the truck was full of hay.

That the amount of hay that ends up on the truck cannot be zero, is supported by the 'unsemanticality' of (135a).

(135) a. ??Alice loaded the truck with no hay.
b. Alice loaded no hay onto the truck.

For *Alice loaded Q hay onto the truck* our entailment wish list is the following.

(136) inferences of: *Alice loaded Q hay onto the truck.*
a. Alice moved Q hay.
b. Q hay ended up on the truck.

It looks like it can have a more simple representation without reference to parts.

(137) Alice moved Q hay with the result that it ended up on the truck.

So it turns out that thematic role properties are not enough to capture telicity related effects, but that decomposition into subevents is necessary. That telic events can be decomposed into a process and a result subevent has been proposed for example by Arsenijević (2006). 'Incremental theme' thus appears to be a derived rather than a primitive notion. Since the problem of argument selection is not my main concern, I will simply assume that Dowty's properties (or something equivalent) can be extracted in some way from the decomposed structures. Decomposition, however, may mean going beyond structural semantics.

The usability of Dowty's properties in semantic representation

We have seen that the property of incremental theme is not directly usable in semantic representations. What about the other properties? Can they be used instead of thematic roles, e.g replacing *agent(e, x)* by *sentient(e, x) & move(x)* or some other combination of properties, depending on the verb, the construction and the participant?

The first problem is that some participants do not seem not to have any of the properties, or for example, only the property of existing independently of the event. This is not restrictive enough. All kinds of things can exist independently of an event without being one of its participants. Engelberg (2004) represents Dowty's roles as follows:

(138) *abtrocknen* 'dry off' (German)
dry(x,y,e) & causer(x,e) & sentient(x,e) & ... & change-of-state(y,e)

But, for reasons mentioned above, we do not want to fix the arity of the verb in this way. An alternative is to introduce a basic relation 'participant', that applies to all participants in addition to possible other properties.

A problem that remains is that it is often debatable whether a participant has a particular property or not. Dowty is careful enough to never give a complete list of properties for any participant.

A second problem is how to prevent *Alice and Bob kissed* from being able to mean that each of them kissed themselves.[4] If there is one event in which Alice and Bob both have exactly the same properties, that boils down to Alice and Bob each being both agent and patient of the event. This means that uniqueness is lost and (139a) would be expected to not only entail (139b), (139c) and (139d), which it does, but also (140a) and (140b), which it does not.

(139) a. Alice and Bob kissed.
b. Alice kissed Bob.
c. Bob kissed Alice.
d. Alice and Bob kissed each other.

(140) a. Alice kissed Alice.
b. Bob kissed Bob.

[4] In some languages there actually is ambiguity between each other and themselves, e.g. German and Italian.

(139b) means: Alice brought it about (volitionally, etc.) that there was a kissing-contact between Alice's lips and Bob (some part of Bob's body).
(139c) means: Bob brought it about (volitionally, etc.) that there was a kissing-contact between Bob's lips and Alice (some part of Alice's body).
(139d) means: Alice brought it about (volitionally, etc.) that there was a kissing-contact between Alice's lips and Bob (some part of Bob's body) and Bob brought it about (volitionally, etc.) that there was a kissing-contact between Bob's lips and Alice (some part of Alice's body) .
These two events may coincide, happening at the same time, with Bob's lips being the relevant part of Bob's body in the first clause and Alice's lips being the relevant part of Alice's body in the second clause. On that reading (139d) means the same as (139a).

Proposed meaning for (139a): Alice brought it about (volitionally, ect.) that there was a kissing-contact between Alice's lips and Bob's lips and Bob brought it about (volitionally, ect.) that there was a kissing-contact between Bob's lips and Alice's lips and the two kissing contacts coincided. In addition, movement is entailed for the kisser's lips (the kisser causes this movement). This makes the kissing contact asymmetrical. A way to do it is (141).

(141) $\exists$e. coinciding_subevents(e) &
$\exists e_1$. kiss(e_1) & agent(e_1, alice) & theme(e_1, Bob's lips) & instrument(e_1, Alice's lips) &
subevent(e, e_1) &
$\exists e_2$. kiss(e_2) & agent(e_2, bob) & theme(e_2, Alice's lips) & instrument(e_2, Bob's lips) &
subevent(e, e_2)

The proposed meaning for (139a) straightforwardly entails (139b), (139c) and (139d). This shows that coinciding subevents are needed for symmetrical predicates. It is necessary to keep each party responsible for their own contribution to the kissing. Therefore uniqueness should hold for each (non-reflexive) subevent: each participant gets only one role per subevent.[5]

Because of the need for different subevents, I conclude that Dowty's properties are important, but not directly applicable for our purposes.

FrameNet

Another interesting approach to thematic roles is the one developed in FrameNet (Baker et al., 1998). Frame semantics describes the meaning of predicates through reference to frames. A frame is intended to characterize a small abstract scene or situation, the properties of which need to be understood in order to understand the semantic structure of a predicate that evokes the frame. A frame also specifies a number of frame elements (FEs) that play a role in the scene described by the frame and can be expressed as satellites of the frame evoker. The names of these frame elements are comparable to

[5] for some predicates, such as *weighs the same as* uniqueness may not be necessary as things anyway weigh the same as themselves

thematic roles. FrameNet uses roles per frame, for example the TRANSFER frame has the roles Donor, Theme and Recipient. These roles are frame specific. Several words instantiate this frame, for example *give* and *receive*. The frame is also used in other, more complex frames. For example, the COMMERCIAL TRANSACTION frame, which has the roles Buyer, Seller, Goods and Money consists of two TRANSFER subevents:

(142)

subev1: the	Buyer gives the	Seller some	Money
	Donor	Recipient	Theme
subev2: the	Seller gives the	Buyer the	Goods
	Donor	Recipient	Theme

These kind of subevents seem to be very similar to the ones I proposed for partially symmetrical predicates like *kiss*. In a semantic representation that supports inference, the more specific roles are probably not necessary. It suffices to use the two subevents with the TRANSFER roles (plus a relation between the events: compensation). The theme in the first subevent can then get the content 'money' or 'value' when it remains unexpressed. If, however, it cannot be maintained that both subevents and all their participants are entailed whenever one of the verbs is used, then the specific roles are needed. I will come back to this discussion in section 3.2.6.

FrameNet is a valuable resource, especially because it is based on corpora. The developers had to assign frame elements to all participants that a frame evoking lexical unit occurs with. They have done so in quite a lot of detail. FrameNet is most developed for English, but even for English it is not complete, neither in lexical and frame coverage, nor in generalizations i.e. relations between frames. The effort of building such a resource is comparable to that of writing a traditional dictionary. A resource containing similar information is currently being developed for Dutch in the Cornetto project (Vossen et al., 2007). What they consider to be their FrameNet component, however, is the Referentie Bestand Nederlands ('Reference Database of Dutch'), which does not seem to contain much of the information that is interesting for us in FrameNet, though this may to some extent be compensated for through their use of WordNet and other ontologies. The D-Coi project (Schuurman and Monachesi, 2006) incorporates FrameNet information into semantic annotation of the Corpus of Spoken Dutch. Padó (2007) suggests that the semantic generalizations that frame semantics makes for English carry over to other languages to a considerable degree. The exact way in which FrameNet information can best be used to aid the choosing of thematic roles in the Delilah lexicon is not clear at present. But it will certainly be useful to look at which distinctions are made. Since FrameNet is based on corpus data, this will help in making sure all possible participants of an event are taken into account. It is also useful to look at relations between frames and their embedding in scenarios, because they will help discover which paraphrases should be accounted for.

FrameNet uses a very large number of roles, or, as they call it, Frame Element names. One would think that Frame Elements (FEs) in more specific frames that are mapped to FEs in more general frames could have the same names as the FEs they are mapped to. One matter that complicates this is multiple inheritance through blending of frames (Fillmore et al., 2004). An example is the blending of the JUDGEMENT frame with the

COMMUNICATION frame. The FEs for the JUDGEMENT frame include the JUDGE, the EVALUEE and the BEHAVIOUR being judged. Verbs that fall in this class are *admire*, *appreciate*, *blame*, *disapprove of*, etc. The COMMUNICATION frame includes (among others) the FEs SPEAKER and ADDRESSEE. Verbs like *praise* and *criticize* can be seen as evoking both these frames. The argument that is the JUDGE in the JUDGEMENT frame corresponds to the SPEAKER in the communication frame[6]. In *flatter*, *compliment*, *scold* the ADDRESSEE is, in addition, equated with the EVALUEE. Since in the *praise* class EVALUEE and ADDRESSEE can be distinct, they cannot be collapsed into one more abstract role.

Also, the FEs of a frame are often not independent of each other. For example a TOPIC (e.g. in the COMMUNICATION frame), tends to be a topic of a MESSAGE. Such relations are sometimes mentioned in the descriptive text, but not formalized.

Some distinctions made are doubtful such as AGENT versus CAUSE (they don't occur together) in the PLACING frame.

(143) a. Alice puts the book into her bag.

b. Clover puts nitrogen into the soil.

Although FrameNet is not (yet) perfect, it explicitly aims at being relevant for inference and since developing a lexicon that supports detailed inference requires a huge lexicographic effort, it would be good if we could use FrameNet or a similar source to improve our lexicon in a (partially) automated way.

The methodology of starting out with verb or frame specific roles (at maximal fine-grainedness every word might have its own frame) and then looking for generalizations seems especially useful.

Van Trijp (2008) did an experiment on the evolution of case marking. Embodied agents played language games in which they had to describe scenes to each other. The introduction of markers was made available as a strategy to avoid ambiguity as to which object introduced by a noun played which role in the event introduced by the verb. Initially, the agents used all case markers that they invented as verb specific. Then, verb specific roles were grouped into more general roles, to be covered by the same case marker, but some specific roles were left over that did not fall into any group. Generalizations emerged on the basis of analogies between actions. The grouping is done on a semantic/conceptual basis, possibly using properties like the ones put forward by Dowty. The purpose is grammatical: It is efficient to re-use grammatical structures. More than one way of grouping is possible. Such generalized roles are likely to support some inferences but lose others. The results of these experiments are thus compatible with Dowty's ideas. The properties he puts forward are then those that happen to be relevant to the grouping reflected in the syntax of English. These most likely are a subset of the ones that are relevant to entailment.

If these experiments are representative for human language, there is no level of generalized semantic roles independent form syntax. In this case, using verb-specific roles and specifying relations between them would be the most principled approach.

[6]For speaker/judge, the participant is first of all speaker; you can compliment/praise without what you are saying corresponding to your actual judgement. You cannot admire someone without meaning it. There is no verb that expresses communicating someone else's judgement.

Some open questions

There appears to be a trade-off between thematic roles and subevents. When we use subevents we need fewer thematic roles. On the basis of what we have seen so far, it seems very unlikely that thematic roles can replace subevents. But maybe subevents can replace thematic roles? Thematic role information may largely or completely follow from decomposition into subevents. Arsenijević (2006) proposes a decompositional analysis of inner aspect and telicity, in which he completely dispenses with thematic roles. Participants are distinguished from each other only on the basis of the subevents they participate in. In the telic template, which is the maximal event structure, there are two subevents; the initiating subevent and the result subevent. There are at most three participants. One participates only in the first subevent, one participates only in the second subevent, and one participates in both.

His approach, however, does not translate straightforwardly to the kind of semantic representation we want to use for inference. A line of future research would be to see if a fine-grained analysis into subevents can make thematic roles superfluous.

A case to illustrate the choice between a dedicated role and a subevent is the question whether the object of *gebruiken* 'use' should get the role of instrument or whether the instrumental preposition *met* 'with' should introduce a *use* event (with its complement as theme) in order to account for the entailments between (144a) and (144b).

(144) a. Alice sneed het brood met een scherp mes.
alice cut the bread with a sharp knife
'Alice cut the bread with a sharp knife.'

b. Alice gebruikte een scherp mes (om het brood te snijden).
alice used a sharp knife (in-order-to the bread to cut)
'Alice used a sharp knife (to cut the bread).'

A problem of letting the adjunct with *met* 'with' introduce an event is that it will be hard to identify the subject of the verb as the agent of this event. A third option is of course that this should follow from world knowledge, or in any case must be stored in a different place than in the lexicon.

Some of Dowty's properties may not follow from other things and yet be important, in particular volition. For example, the difference between *kill* and *murder* seems to be that volition is entailed only for the subject of the latter. That explains why (145a) entails (145b), but not vice versa.

(145) a. Alice murdered Bob.

b. Alice killed Bob.

Is volition assigned in addition to a role? Is it a separate subevent? Or is murdering just a specific type of killing? Evidence against the latter option is that (145a) entails that Alice wanted Bob dead, and (145b) does not.

FrameNet looks like a valuable source of information, as it can be useful to see what distinctions are made there and which frames inherit from each other. I leave it for further research how exactly FrameNet can best be used.

If we do not want to commit to any particulars about the verb meaning, it may also be an option to use grammatical relations instead of thematic roles, i.e. subject, direct object, indirect object, prepositions used, in order to project a coarse semantic structure. We could use a normalization step for passives and nominalizations, and postulates or other sources of lexical knowledge for entailments that do not follow from this structure alone. I do not prefer this approach, because it heavily compromises compositionality, and leaves no room for implicit arguments in the semantic representation.

3.1.4 Conclusions and suggested approach for Delilah

Events are introduced with an existential quantifier, which is kept low in verbs for the most general reading.

On the basis of the discussion on thematic roles, I conclude that we should keep using thematic roles in the semantic representations in Delilah. Also decomposition into subevents is needed. Two such cases we have already seen: Telic predicates need process and resulting state. (Partially) symmetrical predicates need 'parallel' subevents. This leads to analysis below the word level.

Which roles are used is not so important as long as their use is consistent at least between verbs that have some semantic relation. It is best to start out with considering roles to be verb specific.

For determining the roles of the participants of a verb V, I suggest to investigate what entailments hold between different sentences with V, what sentences with a verb less specific than V a sentence with V entails, and possibly also what sentences with a verb more specific than V entail a sentence with V. The latter can probably better be done when studying the more specific verb in question. On the basis of these entailments, it should be determined what roles and subevents are needed to get the entailments. If the list of entailments of a given sentence is not endless, the amount of detail needed to get these entailments is not endless either.

At present, thematic roles are also used for syntactic operations. The passivization rule (involved in the construction of the lexicon) only applies to verbs that assign the theme role to their original direct objects. If we make inference the highest priority when deciding about thematic roles, this may no longer work, because direct objects that licence passive may in some cases get a different role than theme. In this case, the ability to passivize should be based on syntactic features.

In the remainder of this chapter, roles will have traditional names like *agent* and *theme*. It is to be kept in mind that these are relatively arbitrary.

3.2 Events for verbs

This section discusses the implementation of event semantics for different classes of verbs. I take the templates as a starting point, often grouping several of them together, because for example extra NP or PP arguments do not make a difference for the issues discussed. I start with the most simple type of eventive verb, introducing the general machinery. I then move on to auxiliaries. Next, verbs with different types of infinitival

and propositional complements are discussed and after that, I discuss causatives and particle verbs. At the end the implementation of parallel subevents, which were discussed in section 3.1.3, is considered.

3.2.1 Simple eventive verbs

Here, I show the most basic case, verbs introducing a single event and having one or more participants expressed through NPs and PPs. As an example I use the intransitive verb *werken* 'to work' The relevant features of an infinitival entry of *werken* are shown in (146).

(146) *werken* 'to work' (inf.)

```
sem:{store:  {SemSubj binds A,
              λR.∃V. work(V) & event(V) & R binds E},
     body:  λT.(agent_of(E, A) & attime(E, T))}
⋮
node:Top+ID
head:phon:werken
     eventtype:event
     concept:work
synsem:eventvar:E
       external:agent_of∼[Top+ID, A]
⋮
arg:sem:SemSubj
    node:ID+ID2
    synsem:theta:agent_of
```

In the 'body' of the semantics, there is a lambda abstraction over the time. The event argument is only a semantic argument and appears directly in the store. The existential quantifier comes with the event. No mechanism of existential closure has been implemented so far. Such a mechanism will be necessary, if the event variable should be available for different kinds of quantification. Rothstein (1995) proposes, for example, that the expression *every time* actually targets events, and results in universal quantification over the event. The feature `eventtype` distinguishes between events and states. The event variable is being kept track of by the `eventvar` feature, so it can be targeted by adverbs. The variable that the subject binds in the semantics is also kept track of. This is because the infinitive does not syntactically select its subject. The filler of the subject role will be an argument of an auxiliary, modal, raising or control verb.

3.2.2 Auxiliaries and epistemic modals

Auxiliaries and epistemic modals do not introduce an event of their own and do not assign a theta role to their subject either. The subject gets its theta role from the main verb, which occurs in infinitival or participial form. In order for the subject of the inflected verb to be interpreted as the subject of the main verb, the semantics of the subject is recorded under

extsem. This is a technical feature, comparable to the event variable, that is passed on, also by intervening adjuncts, and takes care that the variable is available on every level, so that unification can take place. The same happens, by the way, for the agreement properties of the subject, which also have to be passed on upwards in order to be available when the subject is instantiated. These are not included in the schematic representation, here. The event variable is passed on to the higher level, too. The following example is of an infinitival version of the future auxiliary *zullen* 'will'.

(147) *zullen* 'will/shall' (inf.)

```
sem:{store:  {SemVP applied to T binds V},
     body:  λT.future(V)}
⋮
node:Top+ID
head:phon:zullen
     concept:future
synsem:eventvar:E
       external:Theta∼[Top+ID, A]
       extsem:SemSubj
⋮
arg1:sem:SemSubj
node:ID+ID2
     synsem:theta:Theta
⋮
arg2:sem:SemVP
     node:ID+ID3
     synsem:cat:vp
            eventvar:E
     external:Theta∼[ID+ID3, A]
     extsem:SemSubj
```

Non-epistemic modals pattern with semi-modals in terms of their semantics. These are discussed in the next section.

3.2.3 Infinitival and propositional complements

Control verbs also take infinitival complements and normally bind one of their thematic roles, but, contrary to auxiliaries they do introduce an event of their own and assign a separate theta role to their subject. I discuss perception verbs, verbs with propositional complements, verbs with VP-complements, and object control verbs.

Perception verbs

Verbs of perception take an event as their internal argument. Whether something can occur as the complement of a verb of perception is actually often used to test whether or not it denotes an event.

An implementation like in (148) derives analyses like (149), for perception sentences. (The order of the conjoined predicates does not matter in principle, as long as all variables stay under their quantifier.)

(148) *zien* 'to see' (inf.)

```
sem:{store:  {SemSubj binds A,
              ECM binds B,
              SemVP applied to T binds VP,
              λR.∃V. see(V) & state(V) & R binds E},
     body:  λT. experiencer_of(E, A) & VP & theme_of(E, F)
            & attime(E, T)}
⋮
node:Top+ID
head:phon:zien
     eventtype:state
     concept:see
synsem:eventvar:E
       external:experiencer(Top+ID, A)
⋮
arg:sem:SemSubj
node:ID+ID2
    synsem:theta:experiencer
⋮
arg:sem:SemVP
    node:ID+ID3
    synsem:cat:vp
           eventvar:F
    external:Theta~[ID+ID3, _B]
⋮
arg:sem:ECM
node:ID+ID4
    synsem:theta:Theta
```

(149) Bob zag Alice werken.
Bob saw Alice work
'Bob saw Alice work.'
∃e1. see(e1) & state(e1) & AtPast(t) ∃e2. work(e2) & event(e2) & agent(e2, alice) & attime(e2, t) & experiencer(e1, bob) & theme(e1, e2) & attime(e1, t)

The semantics of the subject of the embedded verb is unified with that of the 'exceptional case marking (ecm)'[7] argument of the matrix verb. The event variable of the embedded

[7]Exceptional Case Marking verbs are those which assign accusative case to the subject of their infinitival complement, without assigning it a thematic role.

verb is used to make it the theme of the matrix verb. The event of the embedded verb is assumed to be instantiable/extensional.

The idea is that in these constructions it is the event of the embedded verb, and not its subject, that is the theme of the matrix verb is supported by the fact that in various languages there seem to be restrictions as to the kind of event that can be the complement of a verb of perception (see also section 3.4.2). Another hint lies in the fact that you can *see it rain*. The expletive does not refer and can therefore not be what is *seen*.

Perceiving an event often goes together with perceiving its participants, but this is not necessarily so. If Alice works at the help desk and controls Bob's computer through remote access to fix a problem for him, while he is watching his screen, then he can see her open a directory without seeing her. Therefore, Alice should not get a role in the matrix event. Also, (150a) seems clearly less contradictory to me than (150b).

(150) a. Alice zag een onzichtbaar spook een pagina uit een boek scheuren.
Alice saw an invisible ghost a page out a book tear
'Alice saw an invisible ghost tear a page out of a book.'

b. Alice zag een onzichtbaar spook.
Alice saw an invisible ghost
'Alice saw an invisible ghost.'

Of course, the helpdesk scenario is an exception and normally it would be useful to be able to conclude from (149) that Bob saw Alice. It is possible to give Alice a theta role in the see event, but the question is which theta role this should be. If it is the same role as the work event, uniqueness of theta roles is violated. If it is a different theta role, that means that if the object of *zien* is a person, it gets a different role than when it is an event.

Verbs of perception may differ from each other in subtle ways. For *horen* 'to hear', the object always needs to be coerced into something audible; a sound. Typically the object produces the sound. From (151a) we may want to infer (151b), but also (151c) and (151d).

(151) a. Alice hoorde Bob de deur open doen.
Alice heard Bob the door open do
'Alice heard Bob open the door.'

b. Alice hoorde Bob.
Alice heard Bob
'Alice heard Bob.'

c. Alice hoorde de deur.
Alice heard the door
'Alice heard the door.'

d. Alice hoorde een geluid.
Alice heard a sound
'Alice heard a sound.'

Giving Bob a separate role in the hearing event would derive the entailment of (151b), but it is not possible to derive the entailment of (151c) in a similar way. I therefore maintain

the analysis that verbs of perception only have two semantic arguments: the perceiver and (the producer of) the stimulus.

Propositional complements

Other control verbs take infinitival arguments that are propositions. They normally have a *that*-clause counterpart. Whether the proposition is stated to be true depends on the nature of the matrix verb. In the example below it cannot be derived to hold true and neither can its negation/opposite. There are also factive and counterfactive verbs.

I assume that both (152a) and (152b) get the representation (152c). The proposition is considered definite (represented by the iota operator), and therefore scope-insensitive. This analysis is backed up by the fact that the complementizer *dat* is clearly related to the definite (demonstrative) article.

(152) a. Bob beweerde te werken.
Bob claimed to work
'Bob claimed to be working.'

b. Bob beweerde dat hij werkte.
Bob claimed that he worked
'Bob claimed that he was working.'

c. ∃e1. claim(e1) & agent(e1, bob) & ιp. proposition(p) & theme(e1, p) & content_of(p, ∃e2 & work(e2) & agent(e2, bob))

The example entry is for the version with an infinitival complement as in (152a).

(153) *beweren* 'to claim' with infinitival complement (inf.)

```
sem:{store:  {SemSubj binds A,
              +SemVP applied to T binds VP,
              λR.∃V. claim(V) & event(V) & R binds E},
     body:  λT. agent_of(E, A) & ιP. proposition(P)
            & theme_of(E, P) & content_of(P, VP)
            & attime(E, T)}
⋮
node:Top+ID
head:phon:beweren
     eventtype:event
     concept:claim
synsem:eventvar:E
       external:agent~[Top+ID, A]
       control:controls(agent~[Top+ID, A],
                        Theta~[ID+ID2, A])
⋮
arg:sem:SemSubj
node:ID+ID1
    synsem:theta:agent
```

```
⋮
arg:sem:SemVP
    node:ID+ID2
    synsem:cat:vp
           eventvar:F
    external:Theta∼[ID+ID3, A]
```

The proposition gets the theme role. This means that it gets the same role as non-clausal (non propositional) direct objects of these verbs do, as in (154a) and (154b).

(154) a. Alice beweerde iets raars.
Alice claimed something strange
'Alice claimed something strange.'

b. Bob zei twee woorden.
Bob said two words
'Bob said two words.'

Some verbs can have propositional complements that do not appear as infinitivals or that-clauses. Examples are *vinden* and *constateren*.

(155) a. Alice vindt Bob aardig / een aardige jongen.
Alilce finds Bob nice / a nice boy
'Alice finds Bob nice / a nice boy.'

b. Alice vindt dat Bob aardig / een aardige jongen is.
Alilce finds that Bob nice / a nice boy is
'Alice finds that Bob is nice / a nice boy.'

c. Alice constateerde griep bij Bob.
Alice diagnosed flu at Bob
'Alice diagnosed Bob with flu.'

d. Alice constateerde dat Bob griep had.
Alice diagnosed that Bob flu had
'Alice diagnosed Bob with flu.'

Here is an example of an entry of *vinden* with an adjectival predicate. The semantics is modeled after the small clause structure (see section 3.2.4).

(156) *vinden* 'find' with small-clause (inf.)

```
sem:{store:  {SemSubj binds A, SemObj binds B
              SemAP applied to B binds SC,
              λR.∃V. find(V) & state(V) & R binds E},
     body:  λT. experiencer_of(E, A) & ιP. proposition(P)
            & theme_of(E, P)& content_of(P, L)
            & attime(E, T)}
⋮
```

```
node:Top+ID
head:phon:vinden
     eventtype:state
     concept:find
synsem:eventvar:E
       external:experiencer~[Top+ID, A]
⋮
arg1:sem:SemSubj
     node:ID+ID1
     synsem:theta:experiencer
            cat:np
⋮
arg2:sem:SemAP
     node:ID+ID2
     synsem:eventvar:F
            cat:adjectival_phrase
⋮
arg3:sem:SemObj
     node:ID+ID4
     synsem:cat:np
```

Nothing has been done here to introduce or bind a temporal argument in the embedded proposition.

VP-complements of 'semi-modals'

Subject control verbs that take what Cremers (1983) identifies as VP-complements, as opposed to the clausal complements above, are harder to represent. In the present implementation the whole event structure of the embedded verb is embedded as the theme of the matrix verb.

Epistemic modals do not have this problem. They are similar to auxiliaries and introduce a modal operator over the whole event structure of the embedded verb. They do not introduce an eventuality of their own.

Another complication is the instantiability of the event of the embedded verb. To some extent this problem also applies to the verbs with propositional complements above. The instantiability is not only dependent on the directly embedding verb but also on the context which that verb is embedded in, such as negation or other verbs of this type. For example, (157a) implies (157b) and (158a) implies (158b). These are (at least in the case of *forget* conventional implicatures, rather than entailments, because they can be cancelled in context. They are however of the computable type. They are clearly and systematically introduced by these particular lexical items.

(157) a. Bob forgot to close the window.
b. $\Rightarrow$ Bob did not close the window.

(158) a. Bob did not forget to close the window.

b. ⇒ Bob closed the window.

The effect, however, is sensitive to focus.

(159) a. Bob did not FORGET to close the window (he left it open on purpose).
b. ⇒ Bob did not close the window.

(160a) is an example where two such verbs (*pretend* and *forget*) are stacked. Peeling off outer layers and changing the sign as required, we can infer (160b) and (160c). In addition, it also seems to be possible to infer (160d).

(160) a. Bob pretended not to have forgotten to close the window.
b. ⇒ Bob forgot to close the window.
c. ⇒ Bob did not close the window.
d. ⇒ Bob pretended to have closed the window.

The implicatures do not always switch between positive and negative, some verbs have no implicature in one of the contexts, such as *refuse to* and *try to*.

Nairn et al. (2006) calculate the instantiability in a top down manner, which may very well be the only way to do it.

Object control

Certain object control verbs, like *force* and *cause*, are causative. Here, in contrast to perception verbs, the subject of the embedded verb does get a role from the matrix verb as well. If Alice forced Bob to work, then she did something to Bob that resulted in him working. One event is taken to cause the other. The implementation of causatives is discussed in the next section. Some other object control verbs are propositional, usually with a modal element (*adviseren* 'advise'). Some may be a combination of propositional and causative (*overtuigen* 'convince/persuade').

3.2.4 Causatives

There are a number of different causative structures. The object control causatives were already mentioned. Related to these are the small clause causatives, based on a transitive verb, as in (161a).

(161) a. Alice verft het hek groen.
Alice paints the fence green
'Alice paints the fence green.'
b. ιx. fence(x) & $\exists$e. paint(e) & agent(e, alice) & theme(e, x) & $\exists$s. green(s) & theme(s, x) & result(e, s)

An analysis like (161b) can be obtained if the adjective introduces state. The motivation for and details of states for adjectives are discussed in section 3.4. The 'result' relation between the event and the state comes in as part of the small clause construction. The variable R in the body is instantiated by the semantics of the adjective applied to the

variable bound by the direct object. Suppose the adjective is *groen*, then `Res` will be ∃*S. green(S) & theme(S, B)*.

(162) *verven* 'paint' selecting a small clause

```
sem:{store:  {SemSubj binds A, SemObj binds B
              SemAP applied to B binds Res,
              λR.∃V. paint(V) & event(V) & R binds E},
     body:  λT. agent_of(E, A) & theme_of(E, B) & Res
            & result_of(E, S) & attime(E, T)}
⋮
node:Top+ID
head:phon:verven
     eventtype:event
     concept:paint
synsem:eventvar:E
       external:agent~[Top+ID, A]
⋮
arg:sem:SemSubj
    node:ID+ID1
    synsem:theta:agent
           cat:np
⋮
arg:sem:SemAP
    node:ID+ID2
    synsem:eventvar:S
           cat:adjectival_phrase
⋮
arg:sem:SemObj
    node:ID+ID4
    synsem:theta:theme
           cat:np
```

The semantics of the adjective could also bind a variable in the stored event quantifier instead, but that would make no real difference. The result state cannot take scope over the event.

I see no reason to postulate a bigger event of which both the event and the state are subevents.

Levin and Rappaport Hovav (1999) discuss two causative structures that differ slightly from each other and argue that causatives with a reflexive, as in (163a), are complex (consist of two subevents), whereas similar causatives without a reflexive, as in (164a) are simple events (event coidentification). They propose the event structures in (163b) and (164b), respectively.

(163) a. Robin danced herself stiff.

b. ∃e1∃e2[Dancing(e1) & Agent(e1,Robin) & Become-Stiff(e2) & Theme(e2,Robin) & Cause(e1,e2)]

(164) a. Robin danced out of the room.
b. ∃e[Dancing(e) & Agent(e,Robin) & Go-Out(e) & Source(e,the room)]

A shortcoming of both these representations, is that they do not entail that Robin ends up outside the room or stiff, respectively. At least they do not do so without extra machinery, such as meaning postulates for *Become-Stiff* and *Go-Out*. And for the case of event coidentification, it is not clear where the go-out event comes from in the composition. It is possible that the verb *dance*, in one of its senses, also has a meaning component that is similar to the meaning of *go* and that this is the sense that combines with a directional PP. In that case the going may not need to be spelled out. A related possibility is that *dance* can occur in a construction that adds the *go* component to its meaning and subcategorizes for a directional PP, such that the event is conceptualized as both *dance* and *go*. A third option is that the verb does not select the directional PP, but that the PP selects a suitable verb to express the event that can realize the movement along the path included in the PP (Gehrke, 2008). All these options can lead to the following alternative representation, in which, for the moment, the internal structure of the result state has not been worked out yet. Arguably, also a path should be introduced.

(165) a. Robin danced out of the room.
b. (∃e)(∃s)[dance(e) & go(e) & agent(e,Robin) & result_of(e, s) & theme(s, Robin) & outside-the-room(s)]

The alternative analysis of (163a) remains close to the original. Only the 'becoming stiff' has been further analyzed into an event with a result state.

(166) a. Robin danced herself stiff.
b. (∃e1)(∃e2)(∃s)[dance(e1) & Agent(e1,Robin) & Cause(e1,e2) & result_of(e2,s) & stiff(s) & Theme(s,Robin)]

3.2.5 Particles

Many Dutch verbs come with separable particles. The same verb can occur with different particles, yielding different meanings (e.g. (167)). The meaning of the verb-particle combination is often not transparent/compositional. Therefore the meaning of the verb-particle combination is entirely attributed to the verb, leaving the particle semantically empty.

(167) a. Alice valt op.
Alice falls up
'Alice attracts attention.'
b. Alice valt af.
Alice falls off
'Alice looses weight.'

It is clear that the meanings are complex, but they are not composed of the meaning of the verb and the meaning of the particle. For the transparent cases it might not be necessary to do it this way. Also the semantics of these predicates in general needs to be looked at more carefully, as the particle often introduces a small clause.

(168) Alice komt terug.
Alice comes back
'Alice comes back.'

(168) means that Alice comes and that the result of that will be that Alice is back. This suggests that such transparent particle verbs could also be treated parallel to the causatives above.

Since the result state is dependent on the event, it has to scope below it.

In the case of particle verbs, it does seem to be useful to introduce an event with subevents. In the small clause construction (e.g. *groen verven* 'paint green') it was clear that the verb introduced the event and that the adjective introduced the state. They did so independently of each other. The state was the result of the event that was contributed by the construction. In particle verbs, the division of tasks is usually not that clear. Take *schoonmaken* 'to clean': does it have as a result that the object is clean? There is some evidence for this. If the object remained dirty, one would say (169a), rather than (169b).

(169) a. Ik heb het proberen/geprobeerd schoon te maken, maar het is nog steeds vuil.
I have it try/tried clean to make, but it is still dirty
'I have tried to clean it, but it is still dirty.'

b. ? Ik heb het schoongemaakt, maar het is nog steeds vuil.
I have it cleanmaked, but it is still dirty
'I have cleaned it, but it is still dirty.'

Notice that in the intended reading of (169a) only the result is intensionally embedded under *proberen* 'to try'. A problem is, that (169b) does not sound like complete nonsense. This may be because the standard for resulting cleanness seems to be to some extent dependent on how clean or dirty the object was before. *Vuil* 'dirty' in the second clause would then be evaluated against more independent standards. The compositional interpretation is to some extent available, as in (170), but it does feel a bit like language play. The English equivalent *to clean* does not support this move. It seems to be a matter of morphology.

(170) (?) Ik heb het schoongemaakt. Nou ja, schoner dan het was.
I have it cleanmaked well cleaner than it was
'I have made it clean. Well, cleaner than it was.'

The lexicalized status of *schoonmaken* also becomes apparent in (171a).

(171) a. Ik moet nog even schoonmaken.
Ik have-to still shortly cleanmake
'I still have to do some cleaning .'

b. * Ik moet nog even vuil maken.
I have-to still shortly dirty make

c. * Ik moet nog even schoonborstelen.
I have-to still shortly clean-brush

d. * Ik moet nog even groen verven.
I have-to still shortly green paint

How should we then represent *schoonmaken*? I think, in this case, it makes sense to have an event with subevents, because *schoonmaken* is also a concept by itself. We would then get a representation as schematically indicated in (172).

(172) ∃e. event(e) & schoonmaken(e) & agent_of(e, subj) & theme_of(e, obj) & ∃p. event(p) & subevent_of(e, p) & agent_of(p, subj) & theme_of(p, obj) & ∃s. state(s) & subevent_of(e, s) & result_of(p, s) & schoon(s) & theme_of(s, obj)

The process that leads to the result cannot be called *schoonmaken* because in this concept the result is already included. The process subevent here is not further specified. It can be made explicit in a PP with *door*, e.g. *door het te borstelen* 'by brushing it'.

Many particle verbs have some sort of resultative structure, but there is quite a lot of variety. In *vastpakken* 'grab, take hold of' the result involves both participants. *vastpakken* results in *vasthouden(/hebben)* 'to hold', whereas *bijkomen* 'to recover' results in the subject returning to an earlier state.

Events consisting of a process and a result subevent are quite widely accepted and there exists some syntactic evidence for their structure. This form of decomposition takes us slightly beyond structural semantics, because it involves decomposition of lexical verbs. It is not clear at present how far we should go with this kind of decomposition.

3.2.6 Parallel sub-events

In section 3.1.3 about thematic roles we have seen two examples in which parallel subevents were proposed. One is the well known *buy/sell* case from FrameNet. Adopting their analysis in our type of semantics is not completely unproblematic. One problem is a lack of evidence for all this structure to be there. Most of the participants cannot be referred to by means of a pronoun if they were not explicitly mentioned. In (173), the pronoun *he* cannot refer to the buyer, if he wasn't mentioned before.

(173) Alice sold her tv. #He was happy with it. (he=the buyer)

This is however also the case for unexpressed underlying subjects of passives (Koenig and Mauner, 1999). In (174) the pronoun cannot refer to the murderer.

(174) Alice was murdered. #He hasn't been caught yet. (he=the murderer)

Yet, it can clearly be inferred from (173) that someone bought Alice's tv, just like it can be inferred from (174) that someone murdered Alice. On the other hand, the acceptability of (175b) suggests that the MONEY participant is not strictly entailed. Yet, there is a certain amount of money that Alice is still entitled to.

(175) a. #Alice sold her tv, but no-one bought it.
b. Alice sold her tv, but she never got the money.

Another point is that the subevents can occur in any temporal order, with an undetermined amount of time in between.

(176) a. Buy now, pay later.
b. Alice bought a new tv, but she doesn't have it yet.

Determining what properties an event needs to have in order to call it *buy* or *sell* clearly falls under lexical semantics. And since lexical meanings are stretchable, it is not possible to define it precisely. Storing the knowledge needed for these inferences elsewhere is likely to involve doing some of the work double, as reference to the structures that can occur will probably be necessary. The choice should ultimately depend on what alternative ways there are to make the relevant information available.

Cases where the subject needs to be split up into different participants are computationally problematic. There must be some mechanism that for intransitive *kiss* (*zoenen* in Dutch) interprets the subject as a group and assigns roles to members of that group. This challenge also applies to the interpretation of explicit reciprocals (Moltmann, 1992).

For the *marry* case, where there is a subject and an object, the fact that the relation also holds the other way around, can quite easily be stored in a wordknowledge component as part of what we know about the meaning of the word.

3.2.7 Discussion and conclusions

This section has presented a basic implementation of event semantics for verbs in Delilah which is a good basis for further refinement. Implementations of event semantics in other deep semantic parsers, such as the ones mentioned in section 2.5 seem to be similarly basic. They typically introduce one event per verb and a number of roles for the participants. In Delilah, the task was approached by starting out from the different templates we have, so that each of our templates now introduces the basic ingredients for an event semantics for the verbs based on it. On a closer look it may turn out that we need more differentiation. For example the template for transitive verbs may need to be split into different templates for transitive verbs corresponding to different aspectual classes.

A general property of the implementation is that main events (from main verbs) are introduced with an existential quantifier in the store of the verb's semantics. Auxiliaries and epistemic modals are assumed not to introduce an event of their own.

Verbs with infinitival and clausal complements are a diverse group. Verbs of perception are a separate type. Their infinitival complement is an event. The subject role filler of this event is treated as an 'ecm' argument of the main verb. When a verb can occur with a clausal complement, this complement is a proposition. The alternating infinitival complement is analyzed in this way as well. There are also some different kinds of complements that can be analyzed as propositional. Propositional arguments get a theta role.

(Semi-)modals are notoriously difficult to represent and need further research. Especially important for these and the verbs with propositional complements are the relative polarity effects that influence entailment. An implementation of these effects, possibly along the lines of Nairn et al. (2006) deserves high priority.

Several causative constructions are covered. Of course, we cannot pretend to be complete here, since a lot of research is still going on in this area. Object control verbs can be causative and/or propositional. Many particle constructions can be considered to have a complex event structure. For some, the particle expresses a result state, but many are less transparent. They will need to be looked at verb by verb. Subevents are not put in the store, as they are not expected to scope outside the main event.

Parallel subevents do not fit very well in the general implementation format, but they can be implemented if needed.

The exact event structure of many verb classes still has to be worked out. However, in the general format, many different kinds of analyses are implementable.

3.3 Nominalizations

This section[8] discusses event semantics for nominalizations and describes the combinatorially relatively comprehensive implementation of lexical nominalizations in Delilah. Nominalizations are nouns that are derived from verbs or adjectives. This section discusses nominalizations derived from verbs. The next section will be about nominalizations derived from adjectives. The focus here is on eventive nominalizations, but see section 3.3.6 for non event-denoting nominalizations. Infinitival nominalizations are in terms of semantics comparable to the lexical ones that we discuss, apart from subtle aspectual differences (Bartsch, 1986). The main difference is that they need some extra syntactic machinery.

The implementation of lexical nominalizations discussed in this section covers the various ways in which semantic arguments can be expressed, including a treatment of support verbs. A novel feature of this implementation is the use of the general pronoun binding algorithm to find possible binders within the sentence for implicit arguments. In the end, I give a short overview of related work for comparison.

3.3.1 Event semantics for nominalizations

The main argument for assuming event semantics for nominalizations, parallel to verbs, are the inference patterns between verbs and their nominalizations. For example, sentence (177a) uses the noun *operatie* ‘operation, surgery’ and (177b) uses the verb *opereren* ‘operate’. The intuition is that (177a) and (177b) are equivalent. They can be inferred from each other.

(177) a. Alice onderging een operatie.
Alice underwent an operation
‘Alice went through/ had surgery.’

[8]The content of this section largely corresponds to that of Reckman and Cremers (2007).

b. Alice werd geopereerd.
Alice was operated
'Alice was operated on.'

The same goes for (178a) and (178b), containing negation.

(178) a. Alice onderging geen operatie.
Alice underwent no operation
'Alice went through/ had surgery.'

b. Alice werd niet geopereerd.
Alice was not operated
'Alice was not operated on.'

Since the narratives in Narrator are about experiences of patients (in the prototype aimed at they are about breast cancer), this kind of information is rather relevant and should preferably not be missed or misinterpreted. If one of the search criteria is, for example, that the narrative should tell about a patient who had surgery, then each of these sentences above, if occurring in a narrative, provides the relevant information to determine whether it meets this search criterion or not. And of each pair, both variants provide the same information.

Opereren and *operatie* introduce the same concept. Also the relation between *opereren*/*operatie* and Alice is the same in both (177a) and (177b). Arguably it can also be inferred in both cases that there is yet someone else involved who is not mentioned, a filler for the agent-slot of *opereren*/*operatie*.

By using neo-Davidsonian event analysis both sentences can be given the same semantic representation. The basic event representation for both (177a) and (177b) is illustrated below. The verb form is taken to name the concept. The verb can be considered as basic in a situation like this because underived nouns do not usually introduce events. Seeing that it does not lie within the scope of this thesis to discuss what is the best way to represent time/tense, the representations are kept very simple in that respect.

(179) $\exists$e.event(e) & operate(e) & agent_of(e, x) & theme_of(e, alice) & at-time(e, past)

For (177b) this kind of representation is quite standard and we have seen in section 3.2 how it is derived. Event representations for event-denoting nominalizations have also been suggested before (Higginbotham, 2000; Parsons, 1990). The verb *ondergaan* in (177a) plays a special role. It places the event in time (makes it extensional) and it lets its subject be the theme of the surgery event.

3.3.2 Expression of participants

The semantic role fillers of the event denoted by the nominalization, which correspond to the arguments of the verb from which it is derived, can be expressed in different ways. I will use the terms 'subject argument' and 'object argument' for what would be respectively the subject and the object of the corresponding verb in an active sentence. Subject and object arguments can by expressed in a prepositional phrase (illustrated in

(180a)), as a prenominal genitive (as in (180b)), as an adjective (as in (180c)), as the first part of a compound noun (shown in (180d)) and through a light verb / support verb construction (example in (180e)). Often, there is ambiguity. For example, (180d) can mean either 'observation by satellite', or 'observation of satellites'.

(180) a. the destruction *of the city by the Romans*
b. *the city's* destruction
c. the *American* invasion
d. *satellite* observation
e. *Bob* made a decision.

Other arguments, such as prepositional or sentential arguments, are expressed in the same way as they were with the verb, except that they are more often optional with nominalizations.

When an argument is not expressed in any of these ways, it can get a referent from the (linguistic or non-linguistic) context, or remain unspecified.

The implementation discussed here is one of Dutch (lexical) nominalizations. Semantic role fillers in Dutch nominalizations surface in essentially the same ways as their English counterparts (Hoekstra, 1999). The Dutch versions of *of* and *by* are *van* and *door*, respectively.[9]

(181) de verwoesting van de stad door de Romeinen
the destruction of the city by the Romans
'The destruction of the city by the Romans'

The prenominal genitive position is more restricted in Dutch than it is in English. Mainly pronouns and proper names occur in this position. In compounding, on the other hand, proper names occur much less easily as the first part of a compound than they do in English.

The implementation in Delilah covers event- and result-denoting nominalizations with semantic arguments expressed as PPs (subject, object and others), as prenominal genitives, as adjectives and as subjects of support verbs. Compounds are covered to a lesser extent. A neo-Davidsonian event semantics is used. Some cases of temporal anchoring of events are covered.

Nominalizations taking complements

Whereas dealing with the optionality of complements to nouns is a major issue for the HPSG based approach by Badia and Saurí (1998) (see section 3.3.8), optionality is not a problem for Delilah. We saw in section 2.2, that when the arguments of a verb occur in different orders or on different sides of the verb, a different lexical item is needed for each configuration, and that the very large lexicon this creates, is not a problem, since it can be searched in very efficiently.

[9] Just like *of*, *van* is also the genitive preposition and just like *by*, *door* also introduces the agents of passives.

It is, therefore, completely in line with the architecture of Delilah to have many different lexical items per nominalization as well. Some of them select PPs as complements. Others do not syntactically realize the fillers of their semantic roles. The (at this level) unexpressed participants are listed as arguments, but lack syntactic category and phonological form. Their semantics is like that of a pronoun. They can be bound later by constituents higher in the structure.

As there are different (sets of) templates for different classes of verbs, such as transitive, intransitive, selecting a PP, etc., we also have different classes of nominalizations, each with its own set of templates. This way we can accommodate peculiarities of particular classes of nominalizations.

For example, a nominalization of a transitive verb, with two argument slots, in principle gets four different templates. One that selects two PPs (with the prepositions *van* and *door*), two that select only one PP, and one without any PPs. Where an argument is not selected in the form of a PP, there is an empty argument slot. The semantics of this slot has the same form as that of a pronoun. It has its normal place in the store of the nominalization's semantics, and binds a variable in the body of the semantics, just like a normal argument would do. The variable that this pronoun binds is made visible for selecting categories. The pronoun can get bound in various ways. If it doesn't get bound, it will simply remain a not further specified pronoun in the sentence semantics.

Below is a schematic representation of the main semantic parts of the entries for *operatie* 'surgery'. The four different versions are summarized into one representation. The paths between brackets are not part of every version.

(182) *operatie* 'operation' (transitive event nominalization)

```
sem:  {store:  {SemSubj binds S, SemObj binds O},
       body:  λE.operate(E) & event(E) & agent(E, S)
              & theme(E, O) (& attime(E, Time))}
⋮
node:Top+ID
head:phon:operatie
synsem:cat:n
       subcat:nominalization
⋮
arg1:node:ID+ID1
     sem:SemSubj
     (cat:pp, head:phon:door)
⋮
arg2:node:(ID+ID2)
     sem:SemObj
     (cat:pp, head:phon:van)
```

Here the abstraction in the body is over the event. This is different from the implementation for verbs, where the event was introduced, already existentially closed, in the store, and the abstraction in the body was over the temporal argument. In nominalizations the event variable gets its quantification from the determiner it is

combined with. The event variable is also the target of predication, for example by a verb that takes the nominalization as an argument. The examples in (183) make it clear that it cannot be the temporal argument that is abstracted over in the body of a nominalization's semantics, because it is the event that was a success, not the time at which it occurred.

(183) a. De operatie was een succes.
the surgery was a success
'The surgery was a success.'

b. Elke operatie was een succes.
every surgery was a success
'Every surgery was a success.'

The temporal argument can be either added here, in the semantics of the nominalization itself, (possibly also bound by a stored element), or it can be added later by a higher predicate. I will not go into the details here.

Theoretically, one could also argue for treating the PPs as adjuncts, so they become the governing category. (Solstad (2007) proposes an analysis along these lines.) Arguments would then uniformly be initially implicit and only one template for the nominal would be needed. Such an implementation is possible, but more complex. As the PPs are semantic arguments, I think it makes more sense to directly interpret them as arguments.

Either way, it is predicted that the event quantifier can never take scope over the participants. This is also what is predicted by May (1985)'s theory of Quantifier Raising. That means that in (184), *elke operatie* should not be able to take wider scope than *een oudere patiënt*. That seems to be a problem.

(184) Elke operatie van een oudere patiënt is extra risicovol.
every operation of a elderly patient is extra risky
'Every surgery of an elderly patient is extra risky.'

I will assume that in the reading where *een patiënt* seems to have narrow scope, it is actually a generic, which is insensitive to scope. On that reading, (184) would be a generalization, similar to (185).

(185) Operaties van oudere patiënten zijn (altijd) extra risicovol.
operations of elderly patients are (always) extra risky
'Surgeries of elderly patients are (always) extra risky.'

Compounds

The implementation does not yet cover noun-noun compounds in a systematic way. If they were written as two separate nouns, like in English, they could be treated as a nominalization taking one of its arguments on its left, in the form of a noun.[10] In Dutch however, compounds are written together as one word. We would therefore need a preprocessing step, that recognizes compounds and splits them up. For the time being,

[10]Note also that this argument is non-referential.

a limited number of compounds is included in the lexicon as nominalizations of which one semantic role is already filled in manually in the semantics of the lexical entry. This is the way in which the less transparent compounds will need to be treated anyway, as collocations, in fact.

Prenominal genitive pronouns

An example of a prenominal genitive pronoun that can be interpreted as the subject or object of a nominalization is found in (186).

(186) Zijn operatie slaagde.
his operation succeeded
'His surgery was successful'

Identifying elements higher in the tree as fillers of semantic roles is not trivial in deep parsing, because the nominalization cannot select them: they are not arguments. Interpreting them is accomplished by making the information about which argument slots are still free and which variable they bind in the semantics visible for selecting categories. The selecting category can then look for this information and use it to select a nominalization that still has a role available for it, and to pick the variable it should bind.

The semantics of the genitive pronoun is unified with the semantics of an unspecified pronoun from an empty argument slot. This way, the pronoun gets person, number, and gender features.

For the argument to be bound it needs to be made available on the higher `synsem` level. We have already seen an example of this kind of lexical binding in control verbs. A point of discussion is whether one external theta-argument is enough in the case of nominalizations. If it is, the implementation can be very much parallel to that of verbs. We would then have active and passive entries for nominalizations. In the active entries, the subject (agent) is the external argument, and in the passive entries the object (theme) is the external argument. In the entry where both roles are already filled there is no external argument (the feature gets the value 'none'). This is different from verbs because there, the basic template from which the entries are derived is based on the infinitive. There is no need to remove the external argument specification in the finite forms, because these will not occur in structures where the external argument is bound from a higher position, that is, they are not selected by auxiliaries or control verbs. A nominalization with all its roles filled through prepositional arguments, however, still occurs as an argument of a determiner, possibly a genitive pronoun. Therefore, such a genitive pronoun should not be tempted to try and bind an argument position that has already been filled. The active/passive distinction would yield one extra entry for a nominalization with two empty slots.

(187) *zijn* 'his' (genitive pronoun)

```
sem:{store:  { SemEv applied to E binds N,
               SemPron binds A},
     body:  λP.ιE. N & P)}
```

```
⋮
head:phon:zijn
node:Top+ID
synsem:cat:np
⋮
arg1:node:ID+ID1
     sem:SemPron
⋮
arg2:node:ID+ID2
     sem:SemEv
     synsem:external:Theta∼[ID+ID2, A]
            cat:n
            subcat:nominalization
```

Examples like (188), however, suggest that one external argument might not be enough, at least if both arguments are to be interpreted as lexically bound by a higher constituent.

(188) Deze chirurg heeft Alice' operatie uitgevoerd.
this surgeon has Alice's operation carried-out
'This surgeon has carried out Alice's surgery.'

In order to allow this, there needs to be a version of the possessive pronoun that specifically selects a nominal with only an empty slot for the object. It binds the theme and returns a category with no empty slots. And then there is a similar version that selects nominals with only an empty subject slot. Then, there are two versions that select nominals with two empty slots; one that binds the subject and returns a constituent which still has an empty object slot, and one that binds the object and returns a constituent which still has an empty subject slot.[11] This boils down to some extra bookkeeping of participant slots.

Solstad (2007) proposes an elegant way to interpret German post-nominal genitives, which can most likely be extended to possessives and prenominal genitives. The genitive has an underspecified relational meaning which can get unified with a participant role. His approach relies on a DRT framework (Kamp and Reyle, 1993). Major adaptations to the organization of Delilah would however be needed to implement this.

Adjectives

Adjectival semantic role fillers (of the type *American invasion*) are treated in a similar way as the genitive pronouns. Only a limited set of adjectives can fill a semantic role of a nominalization. These are linked to the templates that let them select a nominalization and bind one of its free argument slots.

[11] In addition to all this, the possessive pronoun also has its 'normal' possessive version that does is indifferent to empty argument slots in the nominals it selects. When there are free argument slots it simply passes on this information, so it will be visible for the next selecting category (e.g. the verb). Other determiners also do so.

Adjectives that do not fill a semantic role are generally interpreted as event modifiers, parallel to their adverbial counterparts. No extra machinery is needed for this. They apply in the same way as normal intersective adjectives do to simple nouns. They do need to pass on information about the nominalization that is relevant to selecting categories.

Genitive *s*

In a genitive construction with a genitive *s* (example in (189)), the *s* is treated as the head. If the right argument of the genitive *s* is a nominalization, then the left argument, which is a proper name, can bind a free argument slot in this nominalization.

(189) Bobs operatie slaagde.
Bob's operation succeeded
'Bob's surgery was successful.'

(190) genitive *s*

```
sem:{store:  { SemEv binds N, SemName binds A},
    body:  λT.attime(N, T)}
⋮
node:Top+ID
head:phon:s
synsem:cat:np
⋮
arg1:node:ID+ID1
     sem:SemName
synsem:cat:np
       subcat:name
⋮
arg2:node:ID+ID2
     sem:SemEv
     synsem:external:Theta~[ID+ID2,A]
            cat:n
            subcat:nominalization
```

3.3.3 Support verbs

There is a class of verbs that select eventive nominalizations (or other event-denoting nouns) as objects, and then bind one of the arguments with their own subject. Examples of such verbs are *ondergaan* ('undergo') and *uitvoeren* ('carry out').

(191) a. Bob heeft een operatie ondergaan.
Bob has an operation undergone
'Bob went through/ had surgery.'

b. Alice heeft de operatie uitgevoerd.
Alice has the operation carried-out
'Alice has carried out the surgery.'

Whether the subject of the verb binds the agent or the theme of the nominalization, is a lexical property of the verb. For *ondergaan* it is always the theme, and for *uitvoeren* it is the agent. The binding typically is obligatory.

An object binding support verb selects an eventive argument DP that still has an empty object slot, and the subject of the verb binds the object argument of this eventive DP (obligatorily). A subject binding support verb does the same, but then for subject arguments. The information about what argument slots of the nominalization are still free is passed on to the verb by the determiner.[12] Apart from contributing the content of the nominalization's object argument, the verb also contributes the tense specifications to the event in the nominalization. Sentence (192a), with the verb in the past, thus gets the representation in (192b), derived compositionally.

(192) a. Iedereen onderging een operatie.
everyone underwent an operation
'Everyone went through/ had surgery.'

b. $\forall$x.person(x) $\rightarrow$ $\exists$e.operate(e) & agent_of(e, y) & theme_of(e, x) & attime(e, past)

(193) *support verb: ondergaan*

```
sem:{store:  { SemEv binds E, SemSubj binds A},
    body:  λT.attime(E, T)}
⋮
node:Top+ID
head:phon:ondergaan
synsem:cat:vp
⋮
synsem:external:Theta~[Top+ID, A]
⋮
arg1:node:ID+ID1
     sem:SemSubj
⋮
arg2:node:ID+ID2
     sem:SemEv
     synsem:freeobj:Theta~[ID+ID2, A]
            cat:np
```

Since this binding through control is obligatory, and each argument can be expressed only once, an object interpretation of the PP in (194) is ruled out, even though the pronoun

[12] The DP then of course has to select an object argument. In a unification based system, this means one has to add some 'ugly' feature to DP's that do not have an object argument, to prevent them from being selected.

in it can be bound by the subject of the support verb. This corresponds to the intuitions. The PP is in this case interpreted as a general possessive, with an underspecified relational meaning.[13]

(194) Moet Bob die operatie van 'm nog ondergaan?
should Bob that operation of him still undergo
'Does Bob still have to go through this surgery of his?'

When we look at cases with different kinds of quantification over the event, however, we see that the approach above is too simple. For (195a), it gives us the representation in (195b) (all operations of which Alice is the agent occurred in the past), whereas something like (195c) is much more intuitive (all operations were carried out by Alice). That is, the information that Alice is the agent shows up in the restrictor of the universal quantifier, while it should be in the nuclear scope.

(195) a. Alice heeft alle operaties uitgevoerd.
Alice has all operations carried-out
'Alice has carried out all surgeries.'

b. ∀e.operate(e) & agent_of(e, alice) & theme_of(e, x) → at-time(e, past)

c. ∀e.operate(e) & theme_of(e, x) → agent_of(e, alice)

Where the temporal information should be is not immediately clear. It probably should provide a temporal frame for the whole sentence, rather than being either in the restrictor or in the nuclear scope. To get the temporal argument in the restrictor it has to be introduced already in the semantics of the nominalization itself, and then bound at the level of the support verb.

In the present implementation, the temporal argument shows up in the nuclear scope of the event quantifier, rather than in the restrictor because it is introduced at the level of the support verb rather than at the level of the nominalization. Introducing the participants later than they are introduced now, would make analyses like (195c) possible. However, this would mean that participants are only introduced when they are expressed. This is not attractive, because participants can also be bound by discourse, linguistic or non-linguistic. For this, they need to be already present in the representation.

An alternative is to repeat the relevant role in the scope, i.e. in the body of the semantics of the support verb and bind it only there. This yields the representation in (196) for (195a).

(196) ∀e.operate(e) & agent_of(e, y) & theme_of(e, x) → agent_of(e, alice)

It says something like: for every operation with an agent and a theme, Alice was the agent. For this to work, it is important that an event can have only one agent.

With existential quantifiers the separation of restriction and scope is less obvious. On closer inspection, though, it becomes clear that also for existentially quantified events it makes sense to assume that the participant provided by the subject of a support verb occurs in the scope; e.g. there was an operation with an agent and a theme and that agent was Alice.

[13] A strong pronoun in this position would most naturally be interpreted as an agent and not bound.

(197) a. Alice heeft een operatie uitgevoerd.
Alice has an operation carried-out
'Alice has carried out an operation.'

b. ∃e.operate(e) & agent_of(e, y) & theme_of(e, x) & agent_of(e, alice)

Below in (198) is a revised version of the entry for *ondergaan* 'undergo'. I have left the temporal argument as it was. This aspect still needs to be worked on.

(198) *support verb: ondergaan*

```
sem:{store:  { SemEv binds E, SemSubj binds B},
    body:  λT. Theta(E, B) & attime(E, T)}
⋮
node:Top+ID
head:phon:ondergaan
synsem:cat:vp
⋮
synsem:external:Theta~[Top+ID, B]
⋮
arg1:node:ID+ID1
     sem:SemSubj
⋮
arg2:node:ID+ID2
     sem:SemEv
     synsem:freeobj:Theta~[ID+ID2, A]
                    cat:np
```

The theta role, `Theta`, of the argument of the nominalization that is to be bound is identified under `freeobj` (`freesubj` for *uitvoeren*) and copied to the body of the semantics. Note that the variable it binds, `B`, is a new one. The original variable `A` remains unbound (and should be existentially closed).

FrameNet contains quite a range of support verbs for English. As Fillmore et al. (2002) notice, support verbs may add registral, aspectual and other semantic aspects to the predication. Also, many support verbs are collocationally highly restricted. Erbach and Krenn (1993) actually focus on the real idiomatic support verb constructions. But also, for example, *ondergaan* does not just take any kind of eventive nominalization as a complement. However, restrictions on what complements may occur cannot be finitely formulated for open domains.[14] Therefore, no such restrictions have been implemented. For parsing, this is good, because now also novel uses are covered. For generation however, which Delilah also does, it would be useful if support verb constructions can be checked against some information source, such as a corpus. Some attemps have been made to correlate nominalizations with support verbs through extraction from a corpus (Grefenstette and Teufel, 1995).

[14] Stevenson et al. (2004) show that grading the acceptability of support verb - nominalization combinations is a hard task for human annotators

3.3.4 Temporal relations

The events in the nominalizations are temporally dependent on the tense of the verb in the sentence, either directly as in (180e) or (199a), or indirectly, through a temporal preposition as in (199b).

(199) a. The presentation started at three.

b. After the presentation we went for drinks.

In addition, temporal expressions can occur in some of the positions in which arguments can also occur (e.g. *last month's observations*).

Nominalizations that occur with support verbs and other light verbs such as *occur* or *happen*, get their temporal specification from this verb. Typically, this is the only thing the light verb contributes to the semantics, and the temporal information applies to the event argument of the nominalization in exactly the same way as it would to the event argument of a verb.

We also have a tentative implementation for nominalizations that occur in temporal adjuncts, headed by prepositions like *before* and *after*. The event variable of clauses had already been made visible to adjuncts for reasons discussed in 2.3.6. It was therefore decided that the temporal relation would be a relation between two events; the event introduced by the nominalization and the event of the main clause. It would perhaps be more correct to assign a time to the nominalization event, and state that it precedes or follows the time of the event in the main clause. It is certainly possible to do this. The present implementation is only a preliminary one, since the way temporal relations are represented in general is rather limited, and in need of revision.

Except for the inclusion of support verbs, the temporal relations covered in Delilah are comparable to those covered in PUNDIT (Dahl et al., 1987).

Adjuncts like *van gisteren*, the Dutch equivalent of *yesterday's*, are treated as heads, of which the nominalization is an argument. With event-denoting nominals, their interpretation is, therefore, relatively straightforward.

3.3.5 Unexpressed arguments as pronouns

Implicit arguments are treated like (a kind of) pronouns in order to capture most of their binding possibilities within the sentence. An additional advantage of treating implicit arguments like pronouns, is that this way, we avoid the use of free variables, which put the stability of the system at risk. Pronouns are much easier to control.

The mechanism for binding pronouns was explained in section 2.3.8. This same mechanism is also used for implicit arguments, with some adaptations to make sure that arguments that are already lexically bound, are not affected.

The following data show that within the sentence, implicit arguments are optionally bound outside their binding domain in more or less the same fashion as normal pronouns. That binding inside the binding domain is inhibited, corresponds to the fact that an expressed subject cannot bind an unexpressed object (or the other way around), yielding a reflexive interpretation.

In (200a) the pronoun *hij* is most naturally interpreted as binding the implicit theme of *operatie*, in (200b) the pronoun is likely to bind the implicit agent, and in (200c) it may very well not bind either one of the implicit arguments. (The syntactic relations between pronoun and implicit arguments are the same in all three cases. World knowledge determines what is the most likely interpretation.)

(200) a. Na de operatie mocht hij weer bezoek ontvangen.
after the operation was-allowed he again visitors receive
'After the surgery, he was allowed to receive visitors again.'

b. Na de operatie ging hij naar één van z'n andere patiënten.
after the operation went he to one of his other patients
'After the surgery he went to one of his other patients.'

c. Na de operatie mocht hij op bezoek komen.
after the operation was-allowed he on visit come
'After the surgery he was allowed to come for a visit.'

These are the same results we get when the argument to be bound is expressed as a possessive (weak) pronoun. It looks like implicit arguments are bound (within the sentence) under the same conditions as overt pronouns.

(201) a. Na z'n operatie mocht hij weer bezoek ontvangen.
after his operation was-allowed he again visitors receive
'After his surgery, he was allowed to receive visitors again.'

b. Na z'n operatie ging hij naar één van z'n andere patiënten.
after his operation went he to one of his other patients
'After his surgery he went to one of his other patients.'

c. Na z'n operatie mocht hij/zij op bezoek komen.
after the operation was-allowed he/she on visit come
'After his surgery he/she was allowed to come for a visit.'

Under quantification, we get the same effect. The implicit pronoun is bound by the quantifier in exactly the same way as overt pronouns are. The quantificational expression *geen van de patiënten* binds an implicit argument in (202a), an overt pronominal argument of a nominalization in (202b) and a pronominal subject of a verb in (202c).

(202) a. Geen van de patiënten mocht na de operatie direct naar huis.
none of the patients was-allowed after the operation directly to home
'None of the patients was allowed to immediately go home after the surgery'

b. Geen van de patiënten mocht na z'n operatie direct naar huis.
none of the patients was-allowed after his operation directly to home
'None of the patients was allowed to immediately go home after his surgery'

c. Geen van de patiënten mocht nadat hij geopereerd was direct naar huis.
none of the patients was-allowed after he operated was directly to home

'None of the patients was allowed to immediately go home after he had surgery'

There is however one important difference between implicit arguments and pronouns. Whereas pronouns need to be bound at some point in the linguistic or non linguistic context, implicit arguments do not always get bound. This calls for a procedure that checks at a high level, presumably discourse, for unbound implicit arguments and applies low existential closure to these. For example the theme of *operatie* in (203) needs to be existentially closed below the other quantifiers, if it does not get a referent from discourse, because it can co-vary with the surgeons and surgeries.

(203) Elke chirurg heeft een operatie uitgevoerd.
every surgeon has an operation out-carried
'Every surgeon has carried out an operation.'

On the revised analysis of support verb constructions, the agent in the restriction of the event quantifier also needs to be closed.

3.3.6 Events and results

Next to an eventive interpretation, a considerable group of nominalizations also have a result state or result object interpretation. For example, *selection* can refer to a selecting event or to the result of such an event, i.e. the group of objects that were selected. Several attempts have been made in the literature to identify the classes of verbs, of which the nominalizations have one or both of these interpretations available, but none of these appear to satisfactorily cover all the data (Osswald, 2005). Result-denoting nominalizations are therefore entered as separate lemmas in the lexicon and have their own set of templates, because different restrictions may apply to them, and because their semantics is different.

Many authors are only interested in the eventive readings of nominalizations. Meyers et al. (1998), however, also take into account result-denoting nouns as well as patient denoting nouns such as *appointee*. This makes sense because an appointee is someone who has been appointed. This means an appointing event can be inferred. The same goes for resultative nominalizations. If the result exists, the event must have taken place. Therefore, in our representation, an event is included in the semantics of a result-denoting nominalization. It is the result of this event that is the target for quantification and predication, i.e. the lambda abstraction is over the result.

Cases in which the event is not strictly entailed, fall beyond the scope of this thesis.

3.3.7 Restrictions

Several restrictions have been formulated on possible argument configurations. A common one is, that some other argument cannot be expressed if the object is not also expressed. The problem with such restrictions is that they are often not really hard. An object that is considered obligatory, can often remain unexpressed, if it is, in FrameNet terms (Johnson and Fillmore, 2000), a definite null argument. This means that the hearer

is expected to know what the filler of this argument role is. Therefore, these restrictions cannot be correctly implemented on the local level, but only on the level of discourse.

Other restrictions, however, are of a different nature. The prenominal genitive position is not available for other objects than affected objects (Grimshaw, 1990). Therefore nominalizations are marked for their object being affected or not.

And in certain result nominals, such as *uitvinding* 'invention' the result corresponds to what would be the object of the verb *uitvinden* 'invent'. Therefore, the nominalization does not take an object argument in the result reading.

3.3.8 Other computational approaches to nominalizations

This section briefly introduces relevant work that has previously been done in computational linguistics on interpreting nominalizations. The intention is to give an overview of what is covered in other approaches.

Quite a variety of treatments for nominalizations have been proposed and implemented, especially in the domains of statistical analysis and shallow parsing. In the HPSG framework, which is equipped for full semantic interpretation, some proposals for the treatment of certain aspects of nominalizations have been put forward. However, no actual implementation in a working system seems to have been reported.

PUNDIT (Dahl et al., 1987; Palmer et al., 1986) is a highly modular system, consisting of distinct syntactic, semantic and pragmatic components, using a lexicon, a broad coverage grammar of English, semantic verb decompositions, rules mapping between syntactic and semantic constituents, and a domain model. PUNDIT deals with nominalizations in a closed domain. Therefore, domain knowledge can be used, when looking in the context for referents for unexpressed essential roles. This account covers at least noun-noun compounds and nouns with PP complements. The semantic output ignores determiners/quantifiers. A temporal analysis is provided for nominalizations, when the nominalization is either introduced by a temporal preposition (*before*, *after*), or is the subject of a verb like *occur*.

Hull and Gomez (1996) propose an interpretation algorithm that attempts to determine the verbal concept of a nominalization and to fill its thematic roles. The different senses in WordNet, and their restrictions on fillers of semantic roles are used. First of all, a distinction is made between verbal and non-verbal senses of nominalizations. Further disambiguation and semantic role filling only applies to nominalizations used in (one of) their verbal senses. Result-denoting nominalizations are considered non-verbal. Fillers for sematic roles can be found in the noun phrase that also the nominalization is contained in, and in prepositional phrases that follow it. This presumably covers all types of argument expression, except for light verb constructions.

Meyers et al. (1998) describe how NOMLEX (Macleod et al., 1998), a dictionary of nominalizations, can be used in information extraction. NOMLEX attempts to list all possible nominalization based paraphrases of a verbal clause. This means that several configurations, in which the noun and its semantic role fillers can occur, are listed. In a configuration, most arguments are optional, but some can be marked as obligatory (typically the object). Also, nominalizations that do not denote events (such as *appointee*) are included. They also discuss the possibility of including temporal information, that

may be expressed in similar ways as the semantic role fillers. Based on NOMLEX, all possible paraphrases of a sentence should be able to be generated and searched for. This way, they avoid having a separate level of semantic representation. Paraphrases are directly mapped onto each other. They also avoid parsing the texts they want to search in.

Lapata (2000) uses statistic methods to determine, for noun-noun compounds of which the second noun is a nominalization, whether the first noun should be interpreted as subject or object. She uses the NOMLEX and CELEX (Burnage, 1990) dictionaries to identify nominalizations in text. For the subject versus object choice, she uses statistics on which words occur as subjects and objects of the corresponding verbs (plus several smoothing techniques). In Lapata (2002) also context is used; i.e. two words preceding and following the nominalization. She reckons this method can also be applied to adjectival subjects and objects of nominalizations.

Terada and Tokunaga (2003) present a corpus based method of transforming nominalized phrases into clauses for a text mining application. They extract candidate nominalizations, based on their morphological similarities to verbs. Through a system of rules, they assign a series of possible analyses. By checking in what combinations the words involved, occur elsewhere in the corpus, they determine what is the most likely analysis. Their analysis consists of bracketing and assigning labels like 'subject', 'object', 'verb' and 'pp'.

Pradhan et al. (2004) use a machine learning approach to semantic argument parsing, to parse arguments of eventive nominalizations in the FrameNet (Baker et al., 1998) database, resulting in shallow semantic analysis. Annotated example sentences from FrameNet are used for training and testing. Particularly interesting is that they include intervening verbs as a feature, distinguishing between auxiliaries, a small set of light verbs, and other verbs. Also the actual form of the verb is taken into account and the path through the parse tree from constituents that might be semantic role fillers to the verb, in order to recognize subjects of support verbs. To my knowledge this is the only system that covers subjects of support verbs as semantic role fillers (although PUNDIT might be able to cover some of these cases through co-reference resolution).

The Parallel PropBank II (Palmer et al., 2005), in contrast to other annotated corpora and treebanks, includes detailed annotation of nominalizations and their arguments, using event structure, for English and Chinese.

Badia and Saurí (1998) present a treatment of optional complements to nouns in a framework that combines HPSG syntax and the semantic approach of the Generative Lexicon (GL) (Pustejovsky, 1995). Complements to nouns are treated as thematically bound adjuncts. This keeps them available for reference, even when they are not syntactically realized. The scope of the implementation seems relatively limited. The paper only discusses complements to nouns and does not mention how semantic role fillers that show up in prenominal genitive position or those that appear as the subject of a support verb can be identified and interpreted.

Another HPSG proposal (Erbach and Krenn, 1993) only discusses support verb constructions, without explaining how this fits in a broader implementation of nominalizations.

3.3.9 Evaluation and discussion

The Delilah system, so far, covers all cases of lexical binding. Implicit arguments that are not lexically bound remain free and can be bound within the sentence by general principles.

Selecting the least complex SLF as the best analysis (see section 2.3.12) favors argument readings over adjunct readings, because the semantics of an argument unifies with an element already present in the store, whereas an adjunct adds a new element. In the case of nominalizations, this means that a PP or a genitive is preferably interpreted as a semantic argument of the nominalization, rather than getting the 'normal', e.g. possessive, interpretation that it would get with a simple noun. This seems to be correct. However, many other ambiguities are still left unresolved, for example, ambiguities between subject and object readings or between different binding options for a pronoun. Only ungrammatical readings are ruled out (or in fact: not produced).

It would be interesting to see, if the present approach can be combined with a statistical approach in such a way that the best analysis is found. Other than in the current statistical approaches, the outcome would be a full fledged 'deep' semantic analysis.

We very much rely on having a good lexicon, but so do most other approaches. For English, NOMLEX (Macleod et al., 1998), a special lexicon for nominalizations has been developed. WordNet and FrameNet (Baker et al., 1998) as well are important sources of lexical information. In addition, some methods have been proposed that can be of help in creating a lexicon of support verbs (Grefenstette and Teufel, 1995). I think that for precise analysis of nominalizations, one needs such information.

3.3.10 Conclusions

A deep parsing, deep semantics implementation of the interpretation of nominalizations and their arguments was presented. It covers most realizations of semantic role fillers, including subjects of support verbs. It was also shown that arguments of nominalizations that remain implicit can be treated as pronouns and be subject to general mechanisms for pronoun binding within the sentence. The event analysis for nominalizations lets entailments between verbs and nominalizations follow naturally.

3.4 States

Several computational semantics systems have by now implemented a form of event analysis for verbs (Bos et al., 2004; Copestake et al., 2005). There has been much debate on whether it is desirable to assume underlying states, parallel to underlying events. Katz (2000) argues against an underlying state analysis, even for stative verbs, whereas Parsons (2000) is ready to accept an underlying state analysis, even for simple nouns. It is clear that states are more problematic than events.

After having argued in the previous section that verbs and their nominalizations, although being of different grammatical categories, introduce the same concept, I will now discuss whether adjectives and their corresponding abstract nouns do so as

well[15]. I show that underlying states give us the same advantages as underlying events, with respect to recognizing concepts across categories for the purpose of inference, as they reify the predicates. I then discuss an alternative representation for copular expressions by Maienborn (2005), which is based on the conviction that the states in these expressions are ontologically different from eventualities, and I show that this alternative representation has unfavorable consequences for inference. I end with a short note on related adjective-adverb pairs.

3.4.1 Adjectives and nouns

In the previous section we have looked at nominalizations of verbs, and seen that event semantics helps us in getting the right entailments. Now we will look at adjectives and their nominalizations. The following pair of example sentences is at least close to equivalent. Who has an illness, is ill. Who is ill, has an illness (at least one).

(204) a. Alice had een ziekte.
Alice had an illness
'Alice had an illness.'

b. Alice was ziek.
Alice was ill
'Alice was ill.'

One could try to treat 'have an illness' as a kind of collocation and this way have (204a) interpreted as *ill(alice)*. This, however leaves no space in the representation for the determiner, which may vary in form and accordingly in interpretation.

For the pair *boos*/*boosheid*, it is more difficult to come up with two equivalent sentences, for lack of a suitable "support verb". Still we can observe that (205a) entails (205b).

(205) a. Bob probeerde zijn boosheid te verbergen.
Bob tried his anger to hide
'Bob tried to hide his anger.'

b. Bob was boos.
Bob was angry
'Bob was angry'

For Katz, however, stative nominalizations denote either a fact or an extent/degree, but never a state. So (205a) could mean that Bob tried to hide (the fact) that he was angry, or how angry he was, but not the state of his being angry. At least the factive reading seems very intuitive here. It is not clear whether there is also a stative reading. In some other contexts, though, a factive reading is not possible. In (206a) *boosheid* is combined with a durational predicate. (A fact does not have a duration; once a fact, always a fact.) An extent or degree reading does not seem to make a lot of sense either.

[15]The contents of this section were published in Reckman and Cremers (2006)

(206) a. Hun boosheid duurt nooit lang.
their anger lasts never long
'Their anger never lasts long.'

b. Ze zijn nooit lang boos.
they are never long angry
'They never are angry for a long time'

Besides, even if *zijn boosheid* in (205a) does only have a factive reading, how should we represent the content of this fact in such a way that (205b) follows from it and that we faithfully represent the quantifier? (*His anger* is definite.) We can't choose a representation like *angry(bob)*, because of the quantifier. But if we represent it as a noun (with a possessive kind of relation to *Bob*), while still using a traditional representation for (205b), then we lose the entailment. So even when embedded in a fact, reification of the predicate still yields better representations.

These considerations lead us to the following type of representation for sentences like (204b) and (205b).

(207) ∃e.state(e) & ill/anger(e) & theme_of(e, alice/bob) & at-time(e, past)

Interestingly, for the adjective-noun pairs it is not always that clear and systematic which is the basic form. For the verb - noun pairs in the previous section, the verb was always basic and the noun was its nominalization. There are also verbs derived from nouns, but they follow a different pattern. Adjective - noun pairs behave less systematically. In the pair *verdrietig* 'sad' - *verdriet* 'sadness', the adjective seems to be the derived form in Dutch, whereas in English the noun has a nominalizing suffix. And for *boos* 'angry' - *boosheid* 'anger' it is the other way around.

3.4.2 An alternative representation

We have seen that adjectives and their "nominalizations" display the same kind of inference patterns as verbs and their nominalizations, and that reification of the predicate, through postulating an eventuality argument, makes these patterns follow naturally. This reification seems to be the crucial point, though. And since independent evidence for a Davidsonian analysis for statives is kind of shaky, we should investigate whether we really need the full structure. Maienborn (2005) proposes a representation for statives which does involve reification of the predicate, but is different from the Davidsonian event structure representation. In this subsection, I discuss this alternative.

Kimian states

Maienborn (2005) argues for a distinction between Davidsonian states (D-states) and Kimian states (K-states), the latter based on Kim (1998). Examples of verbs introducing D-states are *stand*, *sit* and *sleep*. Examples of verbs introducing K-states are *know*, *hate*, *resemble* and copular expressions. In the latter it is the copula that introduces the K-state.

D-states introduce a normal Davidsonian argument, just like other eventualities. For the K-states, Maienborn shows that, like D-states, they are available to anaphoric

reference and time modification, and therefore they need a referential argument. This referential argument, she argues though, is of a different ontological kind than Davidsonian eventuality arguments. It is of a more abstract nature, similar to facts and propositions. The main argument is their deviant combinatorial behavior. K-state verbs cannot serve as the infinitival complement of a verb of perception (see also examples (213b) and (215a) later in this section), they cannot combine with most adverbials, such as manner adverbs and instrumentals, and neither do they combine with locative modifiers, all of this in contrast with D-states and other eventualities. This brings her to the following (tentative) definition of K-states.

(208) *Kimian states:*
K-states are abstract objects for the exemplification of a property P at a holder x at a time t.

Here are some of Maienborn's (German) examples: (209a), with a D-state, is represented as (209b), and (210a), with a K-state, is represented as (210b). The representations are in a flat DRT notation.

(209) a. Carol schläft.
Carol sleeps
'Carol is sleeping.'
b. $[s^{e}, v \mid \text{sleep}(s), \text{theme}(s, v), \text{carol}(v)]$

(210) a. Carol ist müde.
Carol is tired
'Carol is tired.'
b. $[s^{z}, v \mid s \approx [\text{tired}(v)], \text{carol}(v)]$

The embedded box in (210b) contains the property that is the K-state, and the discourse referent *s* reifies this property.

Introduction of the state by the copula

Engelberg (2005) argues the K-state should not be introduced by the copula, but rather by the post-copula predicate (e.g. an adjective), because attributively used adjectives also show the relevant behavior, without being accompanied by a copula.

A problem that Dölling (2005) points out, also supports this. (211a) should entail (211b) and (211c), but if (211a) is to be represented as one complex state, since there is only one copula, these entailments do not follow without extra postulates.

(211) a. Anna ist eine blonde Frau.
Anna is a blond woman.
b. Anna ist blond.
Anna is blond.
c. Anna ist eine Frau.
Anna is a woman.

I conclude that states are not introduced by the copulas but by the predicates themselves.

Extensionalized participants

Engelberg also shows that it is problematic to put individuals introduced by an NP, under the copula in the box that is introduced by "≈" and presents the 'content' of the state. This is because in that case the state in (212a) (being related to Opus) would be a different one than the state in (212b) (being related to George). And while the states in (212b) and (212d) are the same, if Opus is the tuba player of the Deathtöngue, since the subject is in the outer box and therefore extensionalized over, this is not the case for the states in (212a) and (212c).

(212) a. George is related to Opus.
b. Opus is related to George.
c. George is related to the tuba player of the Deathtöngue
d. The tuba player of the Deathtöngue is related to George.

Identity relations between states get more coherent and intuitive if the content of the box embedded under "≈" is restricted to only the core predicate (e.g. *related(x, y)*).

Now, if Engelberg is right that K-states are not more fine-grained than events and D-states, and the content of the embedded K-state box is in all cases only a core predicate, one can wonder what the advantage of the Kimian style representation still is. For facts and propositions this kind of representation is useful, exactly because the content of a proposition is more than a single predicate; it is a full-fledged proposition, and it makes sense to assign a referential argument to the proposition as a whole. Individuals introduced by NPs in embedded propositions are not extensionalized over. If George said that he is related to Opus and if Opus is the tuba player of the Deathtöngue, it is not entailed that George said that he is related to the tuba player of the Deathtöngue. The main remaining difference between the D-state and K-state representations seems to be that the K-state predicate directly predicates over its argument(s), whereas in D-states this relation is mediated through theta roles. It is not clear why this should be the case.

Entailments between K-state and D-state verbs

Representing K-states in a different format than D-states, also causes another complication in the domain of inference. German *liegen* 'to lie' is a D-state verb, hence the grammaticality of (213a). *Sein* 'to be' and also *sich befinden* 'to be located' are K-state verbs, as shown by the ungrammaticality of (213b).

(213) a. Ich sah das Buch auf dem Tisch liegen.
I saw the book on the table lie
'I saw the book lie on the table.'
b. *Ich sah das Buch sich auf dem Tisch befinden.
I saw the book REFL on the table be-located
'I saw the book be located on the table'

But (214a) entails (214b)[16]. (Not all German speakers seem to like the version with the copula, but with *befinden* (214b) is certainly good.) If these two predicates introduce two very different types of states that require different styles of representation, this entailment is problematic.

(214) a. Das Buch liegt auf dem Tisch.
the book lies on the table
'The book is lying on the table.'

b. Das Buch befindet sich / ist auf dem Tisch.
the book located REFL / is on the table
'The book is (located) on the table'

It is of course conceivable that the verb *liegen* actually introduces two substates, one of which is Kimian. Intuitively positional location verbs (with their complements) such as *liegen* convey two different pieces of information. One of these is the location of the subject (expressed by the complement) and the other one is what kind of position the subject is in (upright or lying flat...). The locational information will have to be the K-state that gets us the entailment. That means that the positional information has to constitute the D-state that saves the construction in (213a).

So far the problem seems fixable, be it at the cost of losing the clear-cut distinction between D-state verbs and K-state verbs. (The positional location verbs *stand*, *sit* and *lie* are actually quite a substantial group within the D-state verb class.) But it gets worse. The verb *to sleep* is a D-state verb and *to be asleep*, being a copula construction, behaves like a K-state expression, as is illustrated below, for Dutch.

(215) a. *Ik zag Carol diep in slaap zijn.
I saw Carol deep(ly) in sleep be
'I saw Carol be fast asleep.'

b. Ik zag Carol slapen.
I saw Carol sleep
'I saw Carol sleep.'

But we can observe that (216a) entails (216b).

(216) a. Carol was diep in slaap.
Carol was deep(ly) in sleep
'Carol was fast asleep.'

b. Carol sliep.
Carol slept
'Carol was sleeping'

Here, it is not plausible that (216a) contains a D-state as well as a K-state, because the presence of this D-state should save (215a).[17] Dölling (2005) makes a similar point.

[16] These examples can be reproduced in Dutch, but there the copula version of (214b) is somewhat marginal.

[17] An anonymous reviewer proposed the representation (1a) for 'Carol was asleep'. Made consistent with the view that a K-state is the exemplification of a property that would be (1b). (Where the property is 'being the

Although the distinction between two groups of statives with different behavior is reasonably convincing, I conclude that in a semantic representation for inference purposes, it does not appear to be a good idea to treat *to sleep* and *to be asleep* as fundamentally different kinds of entities. I therefore prefer to stick to Davidsonian style representations for all states. The differences between the two classes that Maienborn shows are of course real. But as they mainly seem relevant for selectional restrictions, they can probably best be captured as part of the feature structure of the predicates, in a computational system like Delilah. In Delilah, the decision of whether two constituents can combine to form a new one depends on the unifiability of their graphs of features. Here, one can include a feature that says for example that a predicate is "abstract". Verbs of perception, all kinds of adverbials and locative modifiers can then be specified for combining only with concrete predicates. The semantic representation then only needs to contain information that is relevant for inference.

Distinction between D-states and K-states

Dölling (2005) also points out that the distinction between D-states and K-states on the criteria that Maienborn proposes is not entirely unproblematic. He shows through (217) that two predicates that are clearly eventive, because they express a change, cannot occur with a locative modifier. Through (218) he shows that they cannot occur as infinitival complements of perception verbs either. That is, eventive predicates behave here like K-states with respect to properties that were used to argue that K-states are of a different nature than Davidsonian arguments.

(217) a. *Hans wurde (gerade) in Italien 30 Jahre alt.
Hans became (at.the.moment) in Italy 30 years old.

b. *Marias Vertrag lief (gerade) in Deutschland aus.
Maria's contract ran (at.the.moment) in Germany out.

(218) a. *Eva sah Hans 30 Jahre alt werden.
Eva saw Hans 30 years old become.

b. *Eva sah Marias Vertrag auslaufen.
Eva saw Maria's contract run out.

Another verb with incoherent behavior is *sich befinden*. It patterns with *stehen* 'stand' (D-state) in (219) where it is modified by an adverb, and with *sein* 'be' (K-state) in (220), where it is the infinitival complement of a perception verb.

theme of a sleep event')

(1) a. [s | s ≈ [s′, v | [sleep(s′), theme(s′, v), carol(v)]]
b. [s^z, v | s ≈ [s'^e | [sleep(s′), theme(s′, v)], carol(v)]]

With a D-state embedded in a K-state, this looks like an interesting compromise. The main problem with it, is that Maienborn introduces K-states next to D-states in order to derive the different combinatory properties of K-states and D-states from their different ontological status. Now if a K-state embeds a D-state, with the same ontological status as any other D-state, one would expect the embedded D-state to also have the same properties as other D-states, such as being able to have a location. This would make the positing of K-states lose its main advantage.

(219) a. Das Auto stand illegal auf dem Fabrikgelände.
the car stood illegally on the factory premises.

b. Das Auto befand sich illegal auf dem Fabrikgelände.
the car found REFL illegally on the factory premises.

c. ?Das Auto war illegal auf dem Fabrikgelände.
the car was illegally on the factory premises.

(220) a. Alice sah das Auto auf dem Fabrikgelände stehen.
Alice saw the car on the factory premises stand.

b. *Alice sah das Auto sich auf dem Fabrikgelände befinden.
Alice saw the car REFL on the factory premises be-located.

c. *Alice sah das Auto auf dem Fabrikgelände sein.
Alice saw the car on the factory premises be.

This provides an additional argument against a fundamental difference between Davidsonian arguments and Kimian states.

3.4.3 Adjectives and adverbs

Adjectives and adverbs are closely related categories (Broekhuis, 1999). If we assume underlying states for adjectives, we should do so for their adverbial counterparts as well. (This is one of the reasons Katz (2000) does not want underlying states for adjectives.) This is not necessarily problematic, because the German *dabei*-construction which Maienborn uses as a diagnostic for whether a predicate has a referential argument, also appears to work for adverbs. In (221), (found with Google), the *da* in *dabei* refers to *schnell*. This means that *schnell* should introduce a referential argument.

(221) Erstaunlich ist, wie schnell und dabei zuverlässig der neue Mozilla Firebird
amazing is how fast and thereat reliably the new Mozilla Firebird
Seiten darstellt.
web sites displays
'Amazing is, how quickly and reliably the new Mozilla Firebird displays web sites.'

This suggests that the representation for these kinds of adverbs can be similar to the one that I have proposed for adjectives.

On the other hand, the main group of adverbs that also occur as adjectives, are the manner adverbs. Real manner adverbs always modify events, even when they occur as adjectives with nouns. They frequently occur with deverbal nouns, but even a *fast car* is a car that drives fast (can drive fast, typically drives fast). In that sense, the choice we made for adjectives may not force us to do the same for manner adverbs.

Next to the real manner adverbs, there also is a group of de-adjectival adverbs. These normally get a manner-like interpretation too, even though the manner aspect was not part of the meaning of the adjective.

When it comes to adjectives, we do not need states for all adjectives either. States are needed for set denoting/intersective adjectives. They are not needed for e.g. relational (*monthly*, *Davidsonian*, *American*, *wooden*) and modal (*alleged*) adjectives. (For an elaborate classification of Dutch adjectives and adverbs, see Broekhuis (1999).)

3.4.4 Simple nouns

There is not much support for underlying states in simple nouns. An argument against it, is that simple nouns do not seem to introduce any thematic roles. In *Alice's anger* Alice is in a state of anger or being angry, but in *Alice's table* Alice is not in any way involved in whatever state *table* may express. The only thing that could possibly be considered to be a participant in a (being a) table state is the thing itself. But if the thing is a participant in the state the word *table* introduces, it is not clear what the state refers to.

3.4.5 Implementation

Let us start with the adjective in predicative position. This is the type we have already seen in the small clause construction (*groen verven*). The semantics of such an adjective, e.g. *boos* 'angry' without an underlying state looks like (222).

(222)
```
{store:{},
 body:  λX.anger(X)}
```

The simplest way to make it a state is shown in (223). Remember that the variable of the state needs to be identified for future reference.

(223) *boos* 'angry' (predicative adjective)

```
sem:{store:  {},
     body:  λX.∃S.anger(S) & state(S) & theme_of(S, X)}
⋮
head:phon:boos
synsem:cat:ap
       eventvar:S
```

In parallel to the events in verbs, the state can also be introduced in the store, as in (224). The result is the same, except that now it would be theoretically possible to let the state quantifier raise to a higher store.

(224) *boos* 'angry' (predicative adjective)

```
sem:{store:  {λRest.∃E.anger(E) & state(E) & Rest binds S},
     body:  λX.theme_of(S, X)}
⋮
head:phon:boos
synsem:cat:ap
       eventvar:S
```

The adjective in adnominal position can certainly not scope out of the NP. The state is introduced as in (225).

(225) *boos* 'angry' (adnominal adjective)

```
sem:{store:{SemN applied to X binds N},
     body:  λX.N & ∃S.anger(S) & state(S) & theme_of(S, X)}
⋮
head:phon:boos
synsem:cat:ap
       eventvar:S
⋮
arg:sem:SemN
    synsem:cat:noun
```

An adjective in predicative position, combines with a copula. The entry for the copula needs to be adapted to make this work. (226) shows a version of the copula *zijn* 'to be', that combines with a state introducing adjective.

(226) *zijn* 'to be' (copula, inf.)

```
sem:{store:{SemN binds X, SemAdj applied to X binds P},
     body:  λT.P & attime(S, T)}
⋮
head:phon:zijn
synsem:cat:vp
       eventvar:S
⋮
arg1:sem:SemN
     synsem:cat:np
⋮
arg2:sem:SemAdj
     synsem:cat:ap
            eventvar:S
```

Modal copulas like *lijken* 'to seem' also introduce a modal operator over the state. The copula *worden* 'to become' introduces an event that results in the state.

The issue with the temporal argument is like with nominalizations. It is probably more correct to let the state start out with a temporal argument, which can then be bound by, for example, the copula.

State denoting nouns are implemented parallel to event-denoting nouns. They get their participants in largely the same ways.

3.4.6 Conclusions

It was shown that a nice side effect of (neo-)Davidsonian event representations, is that entailment relations between verbs and their nominalizations and between adjectives

and their corresponding nouns follow naturally, without any extra machinery. I have defended the use of a Davidsonian representation for adjectives, by showing that assuming states of different ontological sorts obscures certain inferential relations. My point of view is that semantic representations should only contain information that is needed for inference. Information that is relevant for selectional restrictions should be accommodated elsewhere, where it does not interfere with inference.

Underlying states have been introduced in Delilah for intersective adjectives and the nouns that express the same concept as these. Also stative verbs have underlying states. Simple nouns do not get an underlying state representation.

3.5 Stative light verb constructions

Combining the support verb technique with underlying states, we get a way of dealing with the kind of stative light verb constructions that we have seen in 2.3.9, e.g *honger hebben* 'to be hungry'.

3.5.1 The state

Honger behaves as an abstract noun (similar to mass noun), and can therefore occur as a noun and as an NP. If we assume an underlying state for *honger*, then (227) represents some relevant features of the lexical entry of *honger* as an NP. (Remember that `P` is the partial execution version of `P(S)`.)

(227) *honger* 'hunger' (NP)

```
sem:{store:  { SemArg binds A},
     body:λP.∃S.hunger(S) & state(S)
          & experiencer(S, A) & P}
⋮
node:ID+ID2
head:phon:honger
synsem:external:experiencer(ID+ID2, A)
       cat:np
⋮
arg1:node:ID2+ID3
     sem:SemArg
```

This entry of *honger* features an implicit argument that semantically fills the role of experiencer in the state. Like the arguments of deverbal nominalizations, this argument can be bound not only by the subject of a light verb, but also, for example, by a genitive (*zijn honger* 'his hunger').

3.5.2 The light verb

Relevant features of the entry for *hebben* as a light verb are shown in (228). The subject binds the argument of the complement state through control.

(228) *hebben* 'to have' (light verb, inf.)

```
sem:{store:  { SemState binds S, SemSubj binds A},
     body:  λT.attime(S, T)}
⋮
node:Top+ID
head:phon:hebben
synsem:cat:vp
       external:Theta∼[Top+ID, A]
       control:controls(Theta∼[Top+ID, A],
                        Theta∼[ID+ID2, A])
⋮
arg1:node:ID+ID1
     sem:SemSubj
⋮
arg2:node:ID+ID2
     sem:SemState
     synsem:external:Theta∼[ID+ID2, A]
            cat:np
```

When *hebben* takes *honger* as a complement, that results in the SLF shown in (229).

(229) semantics of *honger hebben*

```
{store:  {store:  { SemArg binds A},
          body:  λP.∃S.hunger(S) & state(S)
                 & experiencer(S, A) & P}binds S,
          SemSubj binds A},
 body:  λT.attime(S, T)}
```

For (230a), this yields the representation in (230b).

(230) a. Alice heeft honger.
Alice has hunger
'Alice is hungry.'

b. ∃e.hunger(e) & state(e) & experiencer_of(e, alice) & at-time(e, past)

I chose here to let the thematic role of the external argument of *hebben* be the same as the role of the external argument of its stative complement. It is also possible to let the light verb introduce its own theta role for its subject. In the semantic representation as I propose it here, however, this role will not surface. It may surface if the light verb is taken to introduce a (sub)event of its own. In the case of *honger krijgen* 'to get/become hungry', the light verb *krijgen* will have to introduce an event of which the state is the result. The subject may get a theta role in this event.

3.5.3 Negation

For *geen honger hebben* we need the noun version of *honger*. The determiner *geen* will bind the state, saying that there is no state of being hungry.

(231) *honger* 'hunger' (noun)

```
sem:{store:  { SemArg binds A},
     body:  λS.honger(S) & state(S) & experiencer(S, A)}
⋮
node:ID+ID2
synsem:external:exp∼[ID+ID2, A]
⋮
arg1:node:ID2+ID3
     sem:SemArg
```

The NP *geen honger* can then serve as a complement of a light verb, just like the NP *honger* above. This way the construction is derived compositionally, resulting in the representation (232b) for (232a).

(232) a. Alice heeft geen honger.
Alice has no hunger
'Alice is not hungry.'

b. ¬∃e.hunger(e) & state(e) & experiencer_of(e, alice) & at-time(e, past)

With *geen*, containing a separable, stored negation, as proposed in section 2.3.4, also split scope effects are accounted for (see also 2.3.9).

3.5.4 Degrees

Now, let us have a look at adjectival modification. Adjectives in this construction tend to modify a degree. The same goes for some determiners, like *weinig* 'few/little'. This indicates that the semantics of *honger* must contain reference to a degree too[18].

No effort has so far been made to implement degrees and degree modification in Delilah. Actually, modifiers of adjectives (e.g. *very*) and comparative constructions are not covered at all, the latter because they involve ellipsis. These are some of the most important gaps in the grammar.

I assume that a degree argument can be added to the representation of *honger*, that is picked out by the degree modifier. (This can be done in a similar way as with the event variable that is modified by adverbials.) A problem is how to apply the positive in absence of modification. Alternatively, the main lambda abstraction can be over the degree. In that case, the way tense is applied has to be changed. If this turns out to be the better option, it raises the question whether the state argument is needed at all. An argument against it is that *geen* would then apply to the degree argument as well. *Bob heeft geen honger* would then be interpreted as: 'There is no degree to which Bob is hungry.' This does not seem to be correct. The zero degree is still a degree. This would then require an analysis of *geen* as a degree modifier, returning the zero degree, or a degree below the standard (assuming that hungry has a minimum standard). The

[18] In principle, the modifier could be taken to introduce the degree as well, but this is less likely, unless perhaps in cases of coercion.

interpretation “There is no state of Bob being hungry to at least the (minimum) standard degree” is intuitively more appealing to me.

3.5.5 Conclusion

The important point however, is that no matter the exact analysis of degree modification, as long as it can be implemented compositionally, the analysis of these light verb constructions will be compositional. This means that they can be interpreted with any determiner and adjective that they occur with, i.e. that the variation in the construction is fully accounted for.

3.6 General conclusions event semantics

Events and states have been implemented for verbs, nominalizations, intersective adjectives, and some abstract nouns. The temporal dimension has been largely abstracted away from. I have assumed that it is a level that can be added, with the things that I have discussed here still holding. Aspect is another thing that has only been partly covered. We have looked into aspectual properties in cases where they were shared by all the words that were based on a particular template. However, different verbs based on the same template, such as transitive verbs, may also fall into different aspectual classes. What has been accomplished is a basic implementation on a template basis. It is open for further refinement, after more detailed study of specific phenomena.

For the nominalizations we have focussed on getting the interpretation of the participants right. Now that the general frame is there, future work can concentrate more on the variation among nominalizations, and investigate which nouns exactly should be interpreted as eventive.

It is clear that many problems still need to be solved and that detailed inferencing requires a large lexicographic effort (or a very advanced way of automatically harvesting and incorporating the required knowledge).

The biggest representational problem is that of thematic roles. The more general problem that returns throughout this chapter, is the decomposition of word meaning. Word meanings cannot be precisely defined. Yet, if something is a hard entailment (agreed upon amongst speakers) and follows from lexical knowledge below the word level, this can be taken as a valid reason for including the knowledge needed to derive this entailment as part of the word meaning, and therefore part of the semantic representation.

Chapter 4

Flat Logical Form

An important ambition in the development of the Delilah parser, is that the semantic representations that are its output will facilitate automated inference, for example for the retrieval of information and/or documents. In this chapter the newly developed Flat Logical Form (FLF) is introduced. This is a semantic output format for Delilah that is expected to make detailed semantics-based inference easier. The introduction of events, which was described in the previous chapter, split up the meaning of a sentence into minimal conjuncts, and thereby prepared the ground for FLF, which can be considered a radical form of 'conjunctivism' (Pietroski, 2006).

In this chapter I argue that first-order logic is not the ideal language for representing natural language meanings. On the one hand, there are important natural language expressions that first-order logic cannot satisfactorily represent. On the other hand, reasoning with first-order logic turned out to be so computationally demanding that many who wanted to build robust systems took recourse to simpler but less expressive forms of logic. FLF offers an alternative that is simple (flat) in structure, but rich in information. Valuable information on quantification and embedding is annotated on the variables in the formula, ready to be used for inference.

Section 4.1 discusses the limitations of using first-order logic for representing linguistic meanings. The rest of the chapter is devoted to FLF. First a general introduction to this way of representation is given. Then the different aspects of FLF are discussed separately. Subsequently, the way entailments are computed on the basis of FLF is discussed, and examples are given of how it is expected to work.

4.1 The limitations of first-order logic representations

Automated inference on logical representations of natural language expressions is attractive because of the level of precision that can be achieved. An important question is what kind of logical representation to use. In theoretical semantics, higher order logics are commonly used. In computational semantics, variants of first-order logic are more popular, because of the availability of increasingly well performing first-order theorem provers and model builders.

Blackburn and Bos (2003) argue that first-order logic can be considered a reasonable approximation to the expressive power needed to deal with natural language semantics,

especially when one allows for a rich ontology. They show that with some flexibility about the kinds of entities that can be used in models, a lot of natural language phenomena, such as modality, tense and aspect, and plurals, can be handled in first order logic. Modalities can be expressed through possible worlds, tense and aspect by making use of events/timepoints/intervals, and plurals by introducing groups as entities together with a member relation. The behavior of the extra predicates that are introduced to help express these phenomena, is regulated by postulates. Ultimately, in some of the cases, postulates would be needed that cannot be formulated in first-order logic. In these cases the representation in first order logic is only an approximation. For the future, they expect developments in the domain of description logic, restricted fragments of first order logic that are typically decidable and allow for more efficient inference. These have, however, less expressive power than first order logic does.

Light and Schubert (1997), on the other hand, propose to extend first order logic to cover a range of similar phenomena. They judge this more effective than enriching the ontology with, for example, possible worlds and propositions because they think that complicates both inferencing and the maintenance of their system. They favor one step lexical inferences. One of their extensions concerns nonstandard quantifiers, an issue that Blackburn and Bos (2003) do not discuss. They propose axioms that capture the monotonicity behavior of such quantifiers.

It can be concluded, that for adequate analysis of many natural language phenomena, the expressiveness of first-order logic appears to be the very minimum. Of course it is possible to do interesting and useful things with less, depending on your aims, but it will not be enough to simulate full language understanding even in a purely linguistic sense. As an aside, note that there is also another side to the mismatch between natural language and first order logic. Not only is it not possible for everything that can be expressed in natural language to be expressed in first order logic, but neither can everything that can be expressed in first order logic be expressed in natural language. For example, in natural language, a quantifier always has a restrictor, whereas in first order logic there is no such requirement. This means that predicate logic does not entertain a privileged relation to semantic interpretation: the set of meanings of a natural language is neither a subset nor a superset of the well-formed and interpretable propositions of predicate logic or their complement. It is at best a helpful tool in describing certain aspects of the relations between concepts and operators in natural language.

Whereas first order logic seems to represent a minimum for expressing natural language, for feasible automated inference it rather represents a maximum. Inference on first-order logic representations is computationally very expensive. Especially when the search space increases, it soon becomes problematic to find a solution in a reasonable amount of time. Shallower forms of semantic representation, on the other hand, that allow for more efficient inference algorithms, tend to ignore quantification and/or related phenomena and therefore compromise considerably on precision. In Delilah, a form of semantic representation that we will call Flat Logical Form (FLF) has been developed. It is designed to allow for more efficient inference than first-order logic does, while still retaining the information on quantification.

4.2 An overview of FLF

Chapter two described the underspecified Quasi Logical Form (QLF) and how LFs were derived from it. In this section we will consider a slightly adapted version of QLF which derives Flat Logical Forms (FLFs), plus LFs in an alternative notation. FLF takes the form of a series of conjoined predicates, in which the quantificational properties are coded on the variables, and is expected to facilitate inference-based retrieval. This section explains how FLF is derived and briefly introduces its components. The different aspects of the formalism are worked out in the next sections.

To get an FLF output, a few changes are made to the semantic system explained in chapter one. In the lexicon, the most important change is to the semantics of quantifiers. (233) shows the semantics of *elke* 'every' as we saw it in chapter one.

(233) *elke* 'every'
{store:{Semarg *applied to* X *binds* P}
body:λQ. $\forall$X.(P $\rightarrow$ Q)}

(234) is the version that contains the information needed for FLF. [1]

(234) *elke* 'every'
{store:{Semarg *applied to* X *binds* P}
body:λQ. $\exists$X. quant(X, every) & P & entails1(X, decr) & Q & entails(X, incr)}

The predicate *quant* has as its first argument a variable and as its second the quantifier that binds it. The existential quantifier over X helps parsing the formula, as the variable X needs to be bound. Also theoretically a special status for the existential quantifier is defendable. Jaspers (2005), for instance, assigns a central role to the existential quantifier as the pivot of the whole quantificational system. *Entails1* encodes the entailment property of the left argument of the quantifier (the restrictor) and *entails* encodes the entailment property on the right argument (the nuclear scope). It can have the values incr(easing), decr(easing) and nonm(onotone). These monotonicity properties tell what entailments are licensed. *Every* is monotone decreasing on its left argument and monotone increasing on its right argument. This means that a sentence with *elke* 'every' entails sentences in which the nominal argument is replaced with one that denotes a subset of the set that the nominal denotes (decreasing:going to a smaller set). It also entails sentences in which the verbal argument is replaced with one that denotes a superset of the set denoted by the original verb (increasing:going to a bigger set).

[1] Without partial execution, these would be:

(1) *elke* 'every'
{store:{Semarg *to be converted against* P}
body:λP.λQ. $\forall$X.(P(X) $\rightarrow$ Q(X))}

(2) *elke* 'every'
{store:{Semarg *to be converted against* P}
body:λP.λQ. $\exists$X. quant(X, every) & P(X) & entails1(X, decr) & Q(X) & entails(X, incr)}

Examples are shown in (235a) an (235b) respectively. Here ⟦blond man⟧is taken as a subset of ⟦man⟧and ⟦does something⟧as a superset of ⟦works⟧(Barwise and Cooper, 1981; Zwarts, 1981).

(235) a. Elke man werkt. ⇒ Elke blonde man werkt.
every man works ⇒ every blond man works
b. Elke man werkt. ⇒ Elke man doet iets.
every man works ⇒ every man does something

(236) gives a simplified semantics for the sentence *elke man werkt* 'every man works' after application of the stores. For clarity, semantics of the verb, which would come with its own quantificational structure, has been abbreviated to a simple predicate.

(236) *elke man werkt* 'every man works'
∃X. quant(X, every) & man(X) & entails1(X, decr) & work(X) & entails(X, incr)

The full structure is here:

(237) *elke man werkt* 'every man works'
∃X. quant(X, every) & man(X) & entails1(X, decr) & ∃E. quant(E, some) & work(E) & event(E) & entails1(E, incr) & agent_of(E, X) & attime(E, T) & tense(E, pres) & entails(E, incr) & entails(X, incr)

From this applied logical form (ALF), which is derived from QLF through applying the stores, the two final logical forms are derived. One, which we call Normal Logical Form (NLF), can be considered a notational variant from the LFs we have seen in the previous chapters:

(238) quant(X,every).[man(X) → quant(E,some).[work(E) & event(E) & agent_of(E,X) & attime(E,T) & tense(E,pres)]]

The additional information about entailment directions, needed for FLF has been removed in NLF. 'quant(X, every)' is equivalent to '∀X'. Another difference is that the temporal variable is now free, open to contextual assignment, and that the notation for tense is different. In the form of NLF, the more traditional output format is preserved next to FLF.

The other output logical form is the FLF:

(239) *FLF of 'Elke man werkt.'*
man(X+decr+every+[]) &
work(E+incr+some+[X]) &
event(E+incr+some+[X]) &
agent_of(E+incr+some+[X],X+incr+every+[]) &
attime(E+incr+some+[X],T) &
tense(E+incr+some+[X],pres)

In FLF, each variable occurrence is locally annotated with information as to:

1. the entailment direction of its predicate's environment

2. the quantificational regime it is bound to

3. the variables its instantiation is dependent upon

This makes it a 4-tuple: (variable + entailment direction + binding quantifier + governors). The entailment property indicates whether the predicate of which the variable is an argument, allows for upward, downward or no entailment with respect to the variable. Note that the entailment property of the variable X above varies with the domain of its quantifier: in the restrictor of the universal quantifier, the variable bound by it allows for downward entailment, in the nuclear scope it gives rise to upward entailment. The specification of 'governing' variables indicates whether a variable is referentially dependent or independent. This is the reflection of scope. It is used for dependence on other quantifiers as well as for dependence on intensional operators, and can block entailment, dependent on the nature of the governor.

So, the first conjunct in the example, *man(X+decr+every+[])*, states that a predicate *man* predicates over the variable *X* in a decreasing environment, under the quantifier *every*, and that it is not (or not in a relevant way) scopally dependent on anything else.

This way, all the information related to quantification and scopal operators is coded directly on the variables, and therefore always locally available.

The remainder of this chapter will discuss the FLF representation and its use in more detail and address some problems. We start with the fourth slot, the scopal dependencies. Then we move on to the second slot, the entailment properties. This is a relatively complex matter, because the effects of the composition of different quantifiers need to be calculated through. A related issue both to scope and entailment properties is negation, which gets a section of its own. Here, several issues related to negation are discussed, such as the implementation of the DeMorgan rules for quantifiers and the split scope effects that we already came across in chapter one. These also bear on the representation of quantifiers in the third slot of the FLF variable. The variable itself in the first slot of the 4-tuple is not discussed separately, as its function is just that of a normal variable, which is bound by the quantifier in the third slot. Then, underspecification in FLF is discussed and disjunction is briefly touched upon. Section 4.7 discusses the basic principles for entailment on FLF and some of the main challenges. Finally, an overview is given of how the proposed solutions are envisaged to be integrated into an entailment strategy and give a somewhat more complex example of entailment, for extra illustration.

4.3 Scopal dependencies

In FLF, scope relations are not encoded in the order of the quantifiers, like in first order logic representations, but instead, scopal dependencies are marked in the fourth slot of the variable 4-tuple. The application of the stores to obtain different scope-readings stays largely the same as described in chapter one, but now the division into scope-sensitive and scope-insensitive quantifiers is used in a different way. The idea is that there are two types of quantifiers. One type can be dependent on other quantifiers and the other type is always independent. Take, for example, the sentence *every man reads a book*. Here, *a book* can either be dependent on *every man*, which corresponds to a narrow

scope reading for *a book*, or independent, which corresponds to a wide scope reading. *Every man*, on the other hand, is never dependent on *a book*, because *every* is not one of those quantifiers that can be dependent. Quantifiers that can be dependent are taken to correspond to dynamic quantifiers. These are the quantifiers that can also bind pronouns across sentence boundaries. Quantifiers that are always independent then correspond to static quantifiers, which can only bind pronouns within the sentence. This distinction was introduced in Dynamic Semantics (Groenendijk and Stokhof, 1991). When a generalized quantifier is dependent on another one, the variable bound by this other one occurs in the fourth position of the quadruple of the dependent quantifier's variable. In (240b), one possible reading of (240a), *some book* is dependent on *every man*, indicated by the X in its fourth position. In (240c) it is independent. The fourth position is a list; it can also contain more than one variable[2]. The dependency relation is transitive and antisymmetric.

(240) a. Elke man leest een boek
every man reads a book
'Every man reads a book.'

b. man(X+decr+every+[]) &
book(Y+incr+some+[X]) &
read(E+incr+some+[X]) &
event(E+incr+some+[X]) &
agent_of(E+incr+some+[X],X+incr+every+[]) &
theme_of(E+incr+some+[X],Y+incr+some+[X]) &
attime(E+incr+some+[X],T) &
tense(E+incr+some+[X],pres)

c. man(X+decr+every+[]) &
book(Y+incr+some+[]) &
read(E+incr+some+[X]) &
event(E+incr+some+[X]) &
agent_of(E+incr+some+[X],X+incr+every+[]) &
theme_of(E+incr+some+[X],Y+incr+some+[]) &
attime(E+incr+some+[X],T) &
tense(E+incr+some+[X],pres)

As explained in section 2.3.3, ambiguity is limited where possible. The layeredness of the derivation is preserved. Only when the full structure of the sentence has been computed, it is determined which derivations of the underspecified semantic representation are sensible.

Not only quantifier scope is handled this way, but also other scopal operators. Take, for example, the adverb *misschien* 'maybe'. The meaning of such an operator is represented in FLF as specifying the evidence for a proposition. It is then the proposition that takes scope. Everything that is dependent on the proposition can no longer be

[2]This aspect of the representation shows parallels with skolemnization, where $\forall x.\exists y.R(x,y)$ is rewritten as $\forall x.R(x,f(x))$.

entailed independently of the proposition. Thus, it cannot be inferred from (241) that every man works.

(241) a. Elke man werkt misschien.
every man works maybe
'Maybe every man works.'

b. proposition(A+nonm+the+[]) &
man(B+decr+every+[A]) &
work(C+incr+some+[A,B]) &
event(C+incr+some+[A,B]) &
agent_of(C+incr+some+[A,B],B+incr+every+[A]) &
attime(C+incr+some+[A,B],D) &
tense(C+incr+some+[A,B],pres) &
evidence(A+incr+the+[],possible)

Entities like 'proposition' replace the intensional indexes as used in Montague (1973), where the idea was put forward that each form of embedding in principle introduces intensionality, something that cannot be expressed in standard first order logic. The proposition creates the separate semantic level that is needed for intensionality. Montague (1973) uses meaning postulates to lift this intensionality in some cases. In FLF too, sometimes embedding can be ignored. For example, factive embedding does not block entailment. The choice for introducing abstract entities is also an extension of the reification strategy that started with the introduction of event semantics. It makes the complement, the intensional domain, as a whole addressable, while still displaying its internal structure, leading to a more analytical semantics. The present implementation is a tentative one. It is not yet clear exactly which abstract entities need to be distinguished and which properties of these need to be specified.

Only dependencies that are relevant for referentiality or intensionality are marked. In sections 4.4.2, where we compute effective entailment directions, and 4.5.2, where we apply the DeMorgan rules for quantifiers, we will see that this is not always enough and discuss modifications.

In sum, FLF is an operator-free conjunction. All logical information is specified as indices on the variables in mainly predefined meta-predicates. It can be seen as a logical description of the sentence, but in contrast to description logic it offers full specification of all relevant logical relations. Also, all quantifiers are represented by descriptions, thus integrating first and higher order forms of quantification for inference. Future research will have to establish how different kinds of operators can be sensibly transformed to fit this format. The case of negation will be discussed in section 4.5.

4.4 Entailment properties

The entailment property indicates whether the predicate of which the variable is an argument, allows for upward, downward or no entailment with respect to the variable, that is, it determines the entailment direction. The entailment property is marked on the

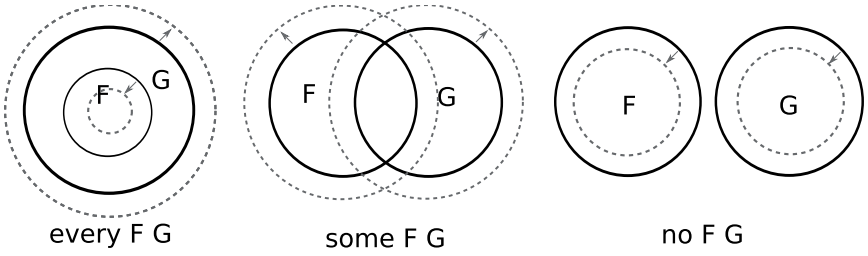


Figure 4.1 — Venn diagrams representing the quantifiers *every*, *some*, and *no*. The arrows and dashed circles indicate the upward and downward entailment directions.

variable, but is relevant for the entire scope of the quantifier. FLF distinguishes between three different entailment properties; monotone increasing, monotone decreasing and non-monotone. Much more fine-grained distinctions can be made. Kas (1993) gives a calculus of how all these functions combine. A calculus of this type can be implemented in FLF. The result would be that some extra inferences can be derived for certain quantifiers.

4.4.1 Increasing and decreasing entailment

Barwise and Cooper (1981) developed the theory of generalized quantifiers for natural language, where determiners can be seen as relations between predicates. The determiner takes a noun as its left argument. This is the first predicate, or set. The determiner and the noun together form a generalized quantifier, a set of sets. The second predicate, the VP, is stated to be a member of this set of sets. The determiner specifies the required overlap between the set denoted by the noun and the set denoted by the VP, for the VP to be a member of the set of sets denoted by the generalized quantifier. For example, *every F G*, with F a noun and G a VP is true iff the set denoted by G is a member of the set of sets of which the set denoted by F is a subset, that is iff F is a subset of G (the intersection of F and G equals F). So *every man works* is true, iff the set of men is a subset of the set of workers. Likewise, *some F G* requires some overlap between the sets F and G, a non-zero intersection, and *no F G* requires no overlap between F and G, a zero intersection. Venn diagrams are given in figure 4.1. Also non-logical quantifiers can be described in this way. For instance, *most F G* requires that the intersection of F and G takes up most of F (e.g. more than half). This way of looking at determiners makes it easy to see certain properties, like monotonicity. Suppose that *every F G* is true, i.e. F is a subset of G. Now if we take a set F′ which is a subset of F, F′ will also be a subset of G and therefore *every F′ G* will also be true. For supersets of F, on the other hand it is not guaranteed that they are still subsets of G. We conclude that *every* is monotone decreasing on its left, nominal, argument. For G, on the other hand, we can take a superset G′. If F is a subset of G, than F is automatically also a subset of G′. If we take G′ a subset of G, however, this is not guaranteed. From this we conclude that *every* is monotone decreasing on its right argument. Monotonicity is often discussed in the context of negative and positive polarity items, which are sensitive to the entailment direction of the environment they

occur in (Zwarts, 1981). Here we purely look at entailment. Monotonicity is defined in terms of entailment, but originates in the determiner. Knowing the entailment properties of the determiner is crucial for being able to derive the correct entailments.

The determiner thus assigns an entailment direction to its left argument (restrictor) and (a possibly different) one to its right argument (nuclear scope). The entailment direction for the restrictor can affect those of other quantifiers in the restrictor (e.g. in a relative clause) and the entailment property for the nuclear scope can affect all other quantifiers in the scope. Note that FLF reflects the theory of generalized quantifiers in an interesting way, as the quantifying determiner forms a natural unit with its (immediate) restrictor: Restricting_predicate(X+Dir+Quant+Dep). (This only holds in simple cases, where no other quantifier is embedded in the restrictor.) The nuclear scope is then indicated through dependency on X. We can indicate scopal dependencies in the fourth slot of the variable quadruple as we have seen in the previous section. Then we get the contrast between (242) and (243) represented as below.

(242) niemand werkt
no-one works

person(A+decr+no+[]) &
work(B+incr+some+[A]) &
event(B+incr+some+[A]) &
agent_of(B+incr+some+[A],A+**decr**+no+[]) &
attime(B+incr+some+[A],C) &
tense(B+incr+some+[A],pres)

(243) iedereen werkt
everyone works

person(A+decr+every+[]) &
work(B+incr+some+[A]) &
event(B+incr+some+[A]) &
agent_of(B+incr+some+[A],A+**incr**+every+[]) &
attime(B+incr+some+[A],C) &
tense(B+incr+some+[A],pres)

Niemand is decreasing on both restrictor and nuclear scope. *Iedereen* is decreasing on its restrictor, but increasing on its nuclear scope. This has consequences for the entailment direction of the event, originally marked as increasing by its own quantifier. In (242), it behaves as decreasing (illustrated in (244a)) and in (243) it behaves as increasing (illustrated in (244b))[3].

[3]Without events it seems to be more straightforward; the verbal predicate predicates over the subject variable, and increasing under negation becomes decreasing, and decreasing under negation becomes increasing. However, more than one place predicates would complicate matters again.

(244) a. No one works.
$\not\Rightarrow$ No one does something.
$\Rightarrow$ No one works hard.

b. Everyone works.
$\Rightarrow$ Everyone does something.
$\not\Rightarrow$ Everyone works hard.

This shows that the entailment direction of dependent predicates cannot be accounted for at the lexical or local level, but that the effective entailment direction needs to be computed, taking into account the quantificational structure of the sentence. In the next few sections, it is discussed how to do these computations.

4.4.2 Properties changing under scope

It is not entirely straightforward to read the relevant entailment property off the representation as it was proposed so far (in the examples above it only appears on one of the arguments of predicate). And actually, it is preferable to only display the resulting entailment direction in the final FLF representation, such that the effective entailment direction is always locally available. In the process deriving this representation, not only the variable of the scopal dependency needs to be marked, but also the relevant entailment property, as below in (245b). This is one of the cases in which it is important to have all (non-increasing) scopal dependencies marked. In (245c) the dependency marker is preserved after its entailment property has been canceled out against the change in the entailment direction of the dependent.

(245) a. niemand werkt
no-one works

b. person(A+decr+no+[]) &
work(B+incr+some+[A+decr]) &
event(B+incr+some+[A+decr]) &
agent_of(B+incr+some+[A+decr],A+decr+no+[]) &
attime(B+incr+some+[A+decr],C) &
tense(B+incr+some+[A+decr],pres)

c. person(A+decr+no+[]) &
work(B+**decr**+some+[A]) &
event(B+**decr**+some+[A]) &
agent_of(B+**decr**+some+[A],A+decr+no+[]) &
attime(B+**decr**+some+[A],C) &
tense(B+**decr**+some+[A],pres)

(1) a. iedereen werkt
person(A+decr+every+[]) & work(A+incr+every+[])

b. niemand werkt
person(A+decr+no+[]) & work(A+decr+no+[])

For explanatory purposes this is displayed here as an operation on FLF, but it would be more correct to see it as a final step in the algorithm deriving FLF. (245b) is an intermediate step on the way to deriving (245c). The entailment properties do not change after this. This way every resulting entailment property is marked locally. The entailment properties being propagated through scope, also means that the ability to create, for example, a decreasing environment is not necessarily reserved for quantifiers/determiners. An example of a non-quantifier influencing the entailment direction of what is in its scope, is negation, which will be discussed in section 4.5.

Now, what we need are the rules for how entailment properties influence each other. To start with, we can use the two theorems from Zwarts (1986):

- If f is monotone increasing and g is monotone decreasing then the composition of f and g is monotone decreasing
- If f and g are both monotone decreasing then the composition of f and g is monotone increasing

We have already seen above that upward entailment changes into downward in the scope of downward entailment, which is consistent with Zwarts. Next, we illustrate how downward entailment influences downward entailment. In (246) *geen man* is downward entailing on *man* and creates a downward entailing environment for its scope. *Elk boek* would normally allow for downward entailment on *boek*, but we see that in the decreasing environment it behaves as upward entailing. *Gelezen* behaves as we have seen before. Originally it would allow for upward entailment. *Elk boek* does not change that, as it does not create a downward entailing environment for its nuclear scope. Being in the scope of *geen man*, however, makes its entailment decreasing. This can be shown by checking whether the sentences in (247) are entailments of (246). For each correct entailment, the conclusion that can be drawn about the entailment direction of the predicate tested is added between brackets, with upward and downward arrows indicating upward and downward entailment, respectively.

(246) Geen man heeft elk boek gelezen.
no man has every boek read
'No man has read every book.'

(247) entailments and non-entailments of (246):

a. $\Rightarrow$ Geen slimme man heeft elk boek gelezen. (man$^{\downarrow}$)
no smart man has every book read

b. $\not\Rightarrow$ Niemand heeft elk boek gelezen.
no one has every book read

c. $\not\Rightarrow$ Geen man heeft elk goed boek gelezen.
no man has every good book read

d. $\Rightarrow$ Geen man heeft alles gelezen. (boek$^{\uparrow}$)
no man has everything read

e. $\Rightarrow$ Geen man heeft elk boek aandachtig gelezen. (lezen$^{\downarrow}$)
no man has every book attentively read

f. $\not\Rightarrow$ Geen man heeft elk boek gezien
no man has every book seen

We see that a decreasing property not only turns upward entailment into downward entailment, but also downward entailment into upward entailment. We can derive the resulting representation as above.

(248) a. Geen man heeft elk boek gelezen.
no man has every boek read
'No man has read every book.'

b. man(X+decr+no+[]) &
book(Y+decr+every+[X+decr]) &
read(E+incr+some+[X+decr]) &
event(E+incr+some+[X+decr]) &
agent_of(E+incr+some+[X+decr],X+decr+no+[]) &
theme_of(E+incr+some+[X+decr],Y+incr+every+[X+decr]) &
attime(E+incr+some+[X+decr],T)

c. man(X+decr+no+[]) &
book(Y+**incr**+every+[X]) &
read(E+**decr**+some+[X]) &
event(E+**decr**+some+[X]) &
agent_of(E+**decr**+some+[X],X+decr+no+[]) &
theme_of(E+**decr**+some+[X],Y+**decr**+every+[X]) &
attime(E+**decr**+some+[X],T)

Now let us see what happens in the scope of two decreasing properties. For that we turn to the (admittedly somewhat unnatural, but presumably true) sentence (249), with double negation.

(249) Geen professor heeft geen boek gelezen.
no professor has no boek read
'No professor did not read any book.'

Also note that (249) is logically equivalent to *every professor read a book*, and that therefore the entailment directions for *professor*, *book* and *read* must be the same in both sentences.

(250) (non-)entailments of (249):

a. $\Rightarrow$ Geen slimme professor heeft geen boek gelezen. (professor$^{\downarrow}$)
no smart professor has no book read

b. $\not\Rightarrow$ Niemand heeft geen boek gelezen.
no one has no book read

c. $\not\Rightarrow$ Geen professor heeft geen goed boek gelezen.
no professor has no good book read

d. $\Rightarrow$ Geen professor heeft niks gelezen. (boek$^{\uparrow}$)
no professor has nothing read

e. $\not\Rightarrow$ Geen professor heeft geen boek aandachtig gelezen.
no professor has no book attentively read

f. $\Rightarrow$ Geen professor heeft geen boek gezien. (lezen$^{\uparrow}$)
no professor has no book seen

What we see is that *gelezen* under two decreasing quantifiers, allows for upward entailment. We can understand that as the same mechanism as above applied twice. Initially, the first scoping quantifier changes the entailment from upward to downward, and then, the second changes it from downward to upward.

(251) a. Geen professor heeft geen boek gelezen.
no professor has no boek read
'No professor did not read any book.'

b. professor(X+decr+no+[]) &
book(Y+decr+no+[X+decr]) &
read(E+incr+some+[X+**decr**,Y+**decr**]) &
event(E+incr+some+[X+decr,Y+decr]) &
agent_of(E+incr+some+[X+decr,Y+decr],X+decr+no+[]) &
theme_of(E+incr+some+[X+decr,Y+decr],Y+decr+no+[X+decr]) &
attime(E+incr+some+[X+decr,Y+decr],T)

c. professor(X+decr+no+[]) &
book(Y+**incr**+no+[X]) &
read(E+**incr**+some+[X,Y]) &
event(E+**incr**+some+[X,Y]) &
agent_of(E+**incr**+some+[X,Y],X+decr+no+[]) &
theme_of(E+**incr**+some+[X,Y],Y+**incr**+no+[X]) &
attime(E+**incr**+some+[X,Y],T)

It is important to notice that the dependencies should not be nested, that is, the directions marked on the variables in the dependency lists do not change in the process. They mark the initial directions of the scoping quantifiers. If the consequences on Y were first calculated through, before considering its influence on E, it would be an increasing property for Y that E was dependent on and we would expect the property on E to end up as decreasing, as in the previous example. It is clear that this is wrong. If we represent it as read(E+incr+some+[X+decr,Y+decr+[X+decr]]) it is immediately visible that the influence of X's quantifier is taken into account twice when computing the effective entailment direction of E's predicate. The computation must be incremental starting at the lowest quantifier. It is easier to see that this is correct in a more traditional representation: $Q_1.P_1(X)\ldots Q_2.P_2(Y)\ldots Q_3.P_3(E)$. Quantifier Q_2 influences the properties of quantifier Q_3, because Q_3 is in the scope of Q_2, then Q_1 influences the properties of Q_2 and Q_3, because they are both in the scope of Q_1.

For natural language quantifiers that can be represented in first order logic (using $\exists$, $\forall$, $\neg\ldots$), the results are the same as would be obtained through logical reasoning. The FLF format also extends to quantification that can not be captured by first order logic. For example, *de meeste N* ('most N') is increasing on its second argument. The following

example by Geurts and van der Slik (2005b), nicely illustrates upward entailment under two non-standard increasing quantifiers.

(252) Most impresarios possess more than seventeen gold watches.
⇒ Most impresarios possess more than seventeen watches.

4.4.3 The influence of non-monotone quantifiers

We have seen that monotone decreasing quantifiers influence the entailment properties of other quantifiers in their scope and monotone increasing quantifiers do not. Now we will look at non-monotone quantifiers. We observe that one non-monotone argument makes the entailment non-monotone. For example, (253) does not entail any of the sentences in (254), nor does (255) entail any of the sentences in (256). Neither the upward nor the downward entailments are preserved. That is, in addition to being sensitive to the scope of negation, entailment properties also need to be sensitive to dependence on non-monotone quantifiers.

(253) precies twee mannen hebben een roman gelezen.
exactly two men have a novel read
'Exactly two men read a novel.'

(254) a. $\not\Rightarrow$ precies twee mannen hebben een boek gelezen.
exactly two men have a boek read

b. $\not\Rightarrow$ precies twee mannen hebben een streekroman gelezen.
exactly two men have a regional novel read

c. $\not\Rightarrow$ precies twee mannen hebben een roman gezien.
exactly two men have a novel seen

d. $\not\Rightarrow$ precies twee mannen hebben een roman aandachtig gelezen.
exactly two men have a novel attentively read

(255) precies twee mannen hebben geen roman gelezen.
exactly two men have no novel read
'Exactly two men did not read a novel.'

(256) a. $\not\Rightarrow$ precies twee mannen hebben geen boek gelezen.
exactly two men have no boek read

b. $\not\Rightarrow$ precies twee mannen hebben geen streekroman gelezen.
exactly two men have no regional novel read

c. $\not\Rightarrow$ precies twee mannen hebben geen roman gezien.
exactly two men have no novel seen

d. $\not\Rightarrow$ precies twee mannen hebben geen roman aandachtig gelezen.
exactly two men have no novel attentively read

If you see the combination of quantifiers as the composition of functions, as Kas (1993) does, (see also 4.4.5), and extend his approach to non-monotone quantifiers, these

observations are not surprising. 'Non-monotone' means 'less than monotone', lacking the property of monotonicity. A non-monotone function is therefore a 'weaker' function than a monotone function, one with fewer special properties that allow for entailments. The composition of a monotone function with a weaker function will yield a weaker function. Entailment patterns like the ones that define monotonicity are only preserved in composition, if they are supported by all composing quantifiers.

We can conclude that in addition to being sensitive to the scope of decreasing quantifiers, entailment properties also need to be sensitive to dependence on non-monotone quantifiers.

Implementing the influence of non-monotone quantifiers in the same way as proposed for decreasing quantifiers leads to a complication, because (253) does entail all of (257) and (255) does entail all of (258).

(257) entailments of (253)

a. ⇒ minstens twee mannen hebben een roman gelezen
at least two men have a novel read

b. ⇒ hoogstens twee mannen hebben een roman gelezen
at most two men have a novel read

c. ⇒ minstens twee mannen hebben een boek gelezen
at least two men have a book read

d. ⇒ minstens twee mannen hebben een roman gezien
at least two men have a novel seen

e. ⇒ hoogstens twee mannen hebben een streekroman gelezen
at most two men have a regional novel read

f. ⇒ hoogstens twee mannen hebben een roman aandachtig gelezen
at most two men have a novel attentively read

g. ⇒ iemand heeft een roman gelezen
someone has a novel read

h. ⇒ iemand heeft een boek gelezen
someone has a book read

(258) entailments of (255)

a. ⇒ minstens twee mannen hebben geen roman gelezen
at least two men have no novel read

b. ⇒ hoogstens twee mannen hebben geen roman gelezen
at most two men have no novel read

c. ⇒ minstens twee mannen hebben geen streekroman gelezen
at least two men have no regional novel read

d. ⇒ hoogstens twee mannen hebben geen boek gelezen
at most two men have no book read

...

If both increasing and decreasing are taken to change into non-monotone in a nonmonotone environment, as illustrated in (259), where (259b) is the intermediate step and (259c) is the result, it is problematic to reconstruct the original direction that is needed for the entailments above.

(259) a. Precies twee mannen hebben een roman gelezen.
exactly two men have a novel read
'Exactly two men read a novel.'

b. man(X+nonm+exactly2+[]) &
novel(Y+incr+some+[X+nonm]) &
read(E+incr+some+[X+nonm,Y]) &
event(E+incr+some+[X+nonm,Y]) &
agent_of(E+incr+some+[X+nonm,Y],X+nonm+exactly2+[]) &
theme_of(E+incr+some+[X+nonm,Y],Y+incr+some+[X+nonm]) &
attime(E+incr+some+[X+nonm,Y],T)

c. man(X+nonm+exactly2+[]) &
novel(Y+**nonm**+some+[X]) &
read(E+**nonm**+some+[X,Y]) &
event(E+**nonm**+some+[X,Y]) &
agent_of(E+**nonm**+some+[X,Y],X+nonm+exactly2+[]) &
theme_of(E+**nonm**+some+[X,Y],Y+**nonm**+some+[X]) &
attime(E+**nonm**+some+[X,Y],T)

Rather than being an entailment property in its own right, non-monotonicity is the lack of an entailment direction. In the case of *exactly n N*, the lack of entailment direction is the result of the quantifier both specifying an upper and a lower limit. *Exactly two N* is equivalent to *at least two N and at most two N*, a coordination of a monotone increasing and a monotone decreasing quantifier, which is therefore non-monotone.

A solution that presents itself here is to represent non-monotone continuous quantifiers already in the lexicon as a coordination of monotone quantifiers. Every continuous function is a finite meet of monotonic functions (Fyodorov et al., 2000). (See section 4.4.5 for a definition of continuous functions.) This is an attractive option, because it makes the entailments more easily available. Its implementability is to be tested in the context of a general treatment of coordination.

A second option, which is also available for discontinuous quantifiers, is to keep (259b) as a final representation, rather than (259c). This reflects the idea that non-monotonicity blocks entailment rather than changing the direction.

An example of a discontinuous environment is the left argument of *most*. Also *most* can entail monotone quantifiers, such as *some*. When this happens the entailment properties of the lower quantifiers become active again.

(260) De meeste mannen die een roman lezen zijn gelukkig.
most men who a novel read are happy
'Most men who read a novel are happy.'

(261) a. $\not\Rightarrow$ De meeste mannen die een boek lezen zijn gelukkig.
most men who a book read are happy

b. $\not\Rightarrow$ De meeste mannen die een streekroman lezen zijn gelukkig.
most men who a regional novel read are happy

c. $\Rightarrow$ Sommige mannen die een roman lezen zijn gelukkig.
some men who a novel read are happy

d. $\Rightarrow$ Sommige mannen die een boek lezen zijn gelukkig.
some men who a book read are happy

Most cannot be analyzed as a combination of monotone quantifiers. An entailment like (261c) must be encoded by a rule that says that an entailment can be produced by replacing most with some. This replacement of course also needs to take effect in the dependencies. That is *P(X+nonm+most+Dep)* is changed into *P(X+incr+some+Dep)* and all occurrences of *X+nonm* into *X(+incr)*.

This is a feasible solution for discontinuous quantifiers. For continuous non-monotone quantifiers decomposition is more attractive, as it straightforwardly accounts for some important entailments, without the use of additional rules.

It is an open question whether this problem also occurs with decreasing quantifiers. The answer depends on whether there are entailments possible in which a decreasing quantifier is replaced with an increasing one, changing the environment for its dependents. This, in turn, depends on the logic to be used (see section 4.7.3). Candidates for such entailments are illustrated in (262). In (262a), *een roman* is in a decreasing environment, created by the determiner *hoogstens twee*. This is shown by the entailment in (262b). But if (262c), where *een roman* is in an increasing enviroment, can be inferred from (262a), then (262d) can be inferred from (262c). It takes a particular (non standard) kind of logic to infer (262c) from (262a) (see also section 4.7.3).

(262) a. Hoogstens twee mannen die een roman lezen zijn gelukkig.
at most two men that a novel read are happy
'At most two men who read a novel are happy.'

b. $\Rightarrow$ Hoogstens twee mannen die een streekroman lezen zijn gelukkig.
at most two men that a regional novel read are happy

c. $\Rightarrow$ Er zijn mannen die een roman lezen.
there are men that a novel read

d. $\Rightarrow$ Er zijn mannen die een boek lezen.
there are men that a book read

With a decreasing effective entailment direction it is in principle easier to calculate back than with a non-monotone one. If you want to replace a decreasing quantifier with an increasing one, it suffices to check all quantifiers that are dependent on it. On these dependent quantifiers, the effect of one decreasing quantifier needs to be subtracted, reconstructing what the property would be without the effect of the relevant decreasing quantifier. This amounts to changing increasing into decreasing and vice versa. It is also possible not to reduce in the final representation, but to always calculate the effect

on-line. But this is not preferred as it makes searching less straightforward, because the same result can be represented in different ways.

4.4.4 Relevance for functional predicates

What we haven't looked at so far, is whether the entailment properties also have consequences for the functional predicates in the representation. Participant roles, for instance, such as *agent_of*, are two-place predicates. The entailment properties do turn out to have some relevance for them. A participant role predicate can be considered upward entailing when a more specific role entails a less specific role, e.g. *agent_of* entails *participant_of*, and downward entailing when a less specific role entails a more specific role, e.g. *participant_of* entails *agent_of*. This presupposes a hierarchy of participant roles. Some evidence for the necessity of such a hierarchy can be found in entailment relations as in (263a) and (263b). In order to explain the entailments and non-entailments one needs to assume that the subject role of *betrokken zijn bij x* 'to be involved in x' is more general than the subject role of *een misdaad plegen* 'to commit a crime'[4].

(263) a. Alice heeft een misdrijf gepleegd. $\not\Leftarrow\Rightarrow$ Alice is betrokken geweest bij een
Alice has a crime committed Alice is involved been at a
misdrijf.
crime

'Alice has committed a crime.' $\not\Leftarrow\Rightarrow$ 'Alice has been involved in a crime.'

b. Geen van ons heeft een misdrijf gepleegd. $\Leftarrow\not\Rightarrow$ Geen van ons is betrokken
none of us has a crime committed none of us is involved
geweest bij een misdrijf.
been at a crime

'None of us has committed a crime.' $\Leftarrow\not\Rightarrow$ 'Non of us has been involved in a crime.'

Such entailments could be accounted for by the relation between the verbal predicates alone, but only if they both assign the same role to their subject. No matter what approach to thematic roles you take, it is unlikely that the predicates in the example at hand can be considered to assign the same roles. (See section 3.1.3 for a discussion of participant roles.)

If we check the inferential behavior of the participant roles in the examples we have seen so far, we can see that they pattern completely with the verb. This is due to the fact that this information is in the scope of the same combination of quantifiers as the event is. We can conclude that in functional predicates that are two-place, the entailment direction that comes with the variable with the lowest scoping quantifier is determining for the entailment direction of the predicate.

[4]If *geen van ons* is replaced with *niemand*, the entailment in (263b) can be considered to hold both ways, because if nobody committed a crime there can not have been one that anyone was involved in, as crimes only exist if someone commits them. This extra entailment is, however dependent on world knowledge, rather than on the entailment properties.

What the entailment property means for the *attime* predicate is unclear, because this predicate does not seem to fit in such a hierarchy. There may be alternative representations of time, where this is different, as there are systematic relations between points or intervals on a time line.

I conclude that functional predicates do not cause problems in the context of entailment directions. To the extent that they fit into a hierarchical ontology, where upward and downward entailment are defined, they behave as expected.

4.4.5 More fine-grained properties of quantifiers

FLF distinguishes between three different entailment properties; monotone increasing, monotone decreasing and non-monotone. Much more fine-grained distinctions can be made. Kas (1993), who views generalized quantifiers as functions, distinguishes discontinuous and continuous functions, where monotone functions are a subset of the latter. This leaves us with two types of non-monotone functions. (In the previous section I argued for decomposition of continuous non-monotone quantifiers.) (264) gives the set theoretical definition of continuous functions, and the natural language inference pattern that holds for them.

(264) continuous:
$f(X \cap Y) \cap f(Y \cup Z) \subseteq f(Y)$
NP VP_1 and VP_2 & NP VP_2 or $VP_3 \Rightarrow$ NP VP_2

Exactly three N is an example of a non monotone, but continuous function. Functions for which this inference pattern doesn't hold are discontinuous (e.g. *more* N_1 *than* N_2). All monotone quantifiers are continuous.

(265) Exactly three princes walk and cry
Exactly three princes cry or talk
- -
Exactly three princes cry

The (maybe) slightly more intuitive pattern in (266) also holds for continuous functions (Fyodorov et al., 2000).

(266) Exactly three princes walk and cry and talk
Exactly three princes talk
- -
Exactly three princes cry and talk
Exactly three princes walk and talk

Monotone quantifiers are either monotone increasing or monotone decreasing. Their inferential properties are listed in (267) an (268), respectively. The first property is the one we have been using so far to show entailment direction, the other two can be derived from it and correspond to two more entailment patterns, relevant for conjunction and disjunction.

(267) monotone increasing:
$X \subseteq Y$ implies $f(X) \subseteq f(Y)$
$f(X \cap Y) \subseteq f(X) \cap f(Y)$ ($>$multi)
$f(X) \cup f(Y) \subseteq f(X \cup Y)$ ($<$add)

NP (VP$_1$ and VP$_2$) $\Rightarrow$ (NP VP$_1$ and NP VP$_2$)
(NP VP$_1$ or NP VP$_2$) $\Rightarrow$ NP (VP$_1$ or VP$_2$)

(268) monotone decreasing:
$X \subseteq Y$ implies $f(Y) \subseteq f(X)$
$f(X) \cup f(Y) \subseteq f(X \cap Y)$ ($<$anti-multi)
$f(X \cup Y) \subseteq f(X) \cap f(Y)$ ($>$anti-add)

(NP VP$_1$ or NP VP$_2$) $\Rightarrow$ NP (VP$_1$ and VP$_2$)
NP (VP$_1$ or VP$_2$) $\Rightarrow$ (NP VP$_1$ and NP VP$_2$)

Two additional properties that a monotone increasing function can have are multiplicativity and additivity. If a function is multiplicative, the inclusion marked above as $>$multi holds both ways, which means that its corresponding entailment pattern is also valid in both directions. The additive property relates to the property marked as $<$add above in the same way.

(269) a. multiplicative:
$f(X \cap Y) = f(X) \cap f(Y)$
NP (VP$_1$ and VP$_2$) $\Leftrightarrow$ (NP VP$_1$ and NP VP$_2$)

b. additive:
$f(X \cup Y) = f(X) \cup f(Y)$
(NP VP$_1$ or NP VP$_2$) $\Leftrightarrow$ (NP (VP$_1$ or VP$_2$)

The counterparts of these properties for monotone decreasing functions are anti-multiplicativity and anti-additivity.

(270) a. anti-multiplicative: $f(X \cap Y) = f(X) \cup f(Y)$
(NP VP$_1$ or NP VP$_2$) $\Leftrightarrow$ (NP (VP$_1$ and VP$_2$)

b. anti-additive: $f(X \cup Y) = f(X) \cap f(Y)$
NP (VP$_1$ or VP$_2$) $\Leftrightarrow$ (NP VP$_1$ and NP VP$_2$)

Two more properties that monotone increasing and decreasing functions can have are consistency and completeness. In order to be consistent it needs to be (anti-)multiplicative or (anti-)additive. In order to be complete it needs to be both.

(271) a. consistent:$f(-X) \subseteq -f(X)$
NP (do) not VP $\Rightarrow$ It is not the case that NP VP

b. complete:$-f(X) \subseteq f(-X)$
It is not the case that NP VP $\Rightarrow$ NP (do) not VP

On the basis of these properties we can distinguish a number of subclasses of monotone functions, next to the basic ones[5].

(272) classes of monotone increasing functions with additional properties:

1. multiplicative
2. additive
3. purely multiplicative (multiplicative + consistent)
4. (purely additive (additive + consistent))
5. homomorphism (multiplicative + additive + consistent + complete)

(273) classes of monotone decreasing functions with additional properties:

1. anti-multiplicative
2. anti-additive
3. (purely anti-multiplicative (anti-multiplicative + consistent))
4. purely anti-additive (anti-additive + consistent)
5. antimorphism (anti-multiplicative + anti-additive + consistent + complete)

In addition, all monotone increasing functions have the 'hidden' properties $<$anti-add and $>$anti-multi, and all monotone decreasing functions have the 'hidden' properties $>$add and $<$multi.

Kas gives a calculus of how all these functions combine. Some of the results are somewhat surprising. Kas's calculus predicts that our more coarse-grained system yields wrong results in some cases. The combination of a quantifier that is anti-additive, anti-multiplicative, or simple monotone decreasing with a simple monotone decreasing or simple monotone decreasing quantifier should not be monotone increasing or decreasing, respectively, but a weaker function. The data do not seem to support this. *Niemand* is anti-additive, *hoogstens n N* is monotone decreasing and *minstens n N* is monotone increasing. Still in (274a) through (276a), the (a) sentences intuitively entail the (b) and (c) sentences.

(274) a. Niemand heeft minstens twee romans gelezen.
no-one has at least two novels read
'No-one read at least two novels.'

b. Niemand heeft minstens twee streekromans gelezen.
no-one has at least two regional novels read
'No-one read at least two regional novels.'

c. Niemand heeft minstens twee romans grondig gelezen.
no-one has at least two novels thoroughly read
'No-one thoroughly read at least two novels.'

[5]The classes between brackets are not attested among natural language quantifiers.

(275) a. Niemand heeft hoogstens twee romans gelezen.
no-one has at most two novels read
'No-one read at most two novels.'

b. Niemand heeft hoogstens twee boeken gelezen.
no-one has at most two regional novels read
'No-one read at most two regional novels.'

c. Niemand heeft hoogstens twee romans gezien.
no-one has at most two novels thoroughly read
'No-one thoroughly read at most two novels.'

(276) a. Hoogstens twee mannen hebben minstens drie romans gelezen.
at most two men have at least three novels read
'At most two men read at least three novels.'

b. Hoogstens twee mannen hebben minstens drie streekromans gelezen.
at most two men have at least three regional novels read
'At most two men read at least three regional novels.'

c. Hoogstens twee mannen hebben minstens drie romans grondig gelezen.
at most two men have at least three novels thoroughly read
'At most two men thoroughly read at least three novels.'

Kas's reasoning in a case like (276a) goes as follows (<additivity is a property of monotone increasing functions and >anti-additivity is a property of monotone decreasing functions):

(277) The composition of f = <additive and g = >anti-additive $(g \circ f)$

1. $g(X \cup Y) \subseteq g(X) \cap g(Y)$ definition >anti-additivity
1a. $g(X \cap Y) \supseteq g(X) \cup g(Y)$ inferred: <anti-multiplicativity
2. $f(X \cup Y) \supseteq f(X) \cup f(Y)$ definition <additivity
2a. $f(X \cap Y) \subseteq f(X) \cap f(Y)$ inferred: >multiplicativity

3. $g(f(X \cap Y))$ hypothesis, lefthand-side 2a.
4. $\subseteq g(f(X) \cap f(Y))$ according to 2a.
5. stuck: 4. matches with the lefthand-side of 1a., but the derivation cannot be continued from here, since the inclusion points in the wrong direction.

Alternative:

3'. $g(f(X)) \cup g(f(Y))$ hypothesis, righthand-side 1a.
4'. $\subseteq g(f(X) \cap f(Y))$ according to 1a.
5'. stuck: 4. matches with the righthand-side of 2a., but the derivation cannot be continued from here, since the inclusion points in the wrong direction.

A possible explanation for the discrepancy between this reasoning and the observations above is that the logic Kas bases his reasoning on may not be the most suitable logic for

capturing natural language intuitions about entailment. In section 4.7.3, the possibility of using a different logic is discussed.

Another issue that comes to mind when studying Kas is that he shows that many verbs behave as homomorphisms, which does not correspond to the existential quantifier that we represent them with. However, not all classes of verbs are homomorphisms. Letting the quantifier match the behavior, would mean having different quantifiers for different verbs.

Kas's calculus, or a variant thereof, can in principle be implemented. The set of entailment properties, then, needs to be extended. Quantifiers are taken to compose when one scopes over the other with nothing intervening. When a quantifier is dependent on more than one other quantifier, the property of the lowest of these needs to take effect first, because the order of composition matters in some cases. The calculus would have to be translated into a set of rules on how the different properties influence each other.

The result of implementing the more fine-grained properties and a calculus for their composition would be that some extra inferences can be derived for certain quantifiers. For the continuous property, which many quantifiers have, this would mean that when trying to derive a proposition of the form NP_{cont} VP, the inference mechanism can search for occurrences of this NP in the two patterns that together allow for this inference. The (anti-) additive and multiplicative properties make extra entailment patterns involving conjunction and disjunction available. Consistency and completeness concern the scope of negation. I will come back to these in the section about negation.

For the time being, we stick to the more coarse-grained approach.

4.4.6 Summary

I showed how the effective entailment direction for a predicate can be calculated. It starts off with the direction it gets from its quantifier. The original direction is influenced by the properties of the other quantifiers it is dependent on. Changes apply according to the schema in (278), as this set of rules turns out to match intuitions best.

(278)
increasing	*under* decreasing	*becomes* decreasing
decreasing	*under* decreasing	*becomes* increasing
increasing	*under* non-monotone	*becomes* non-monotone
decreasing	*under* non-monotone	*becomes* non-monotone

The property of upward monotonicity does not cause any changes to the environments dependent on it. Dependency on an increasing quantifier is therefore not explicitly marked as such. Non-monotonicity is not susceptible to change.

It has advantages for inference if continuous non-monotone quantifiers are represented as a combination of monotone ones. Only discontinuous quantifiers are then still marked as non-monotone.

4.5 Negation

Being able to handle negation is a crucial test for FLF. We will show that also negation needs to be described in a flat way and not used as a logical operator in the structure at the same level as the conjunction.

4.5.1 Flattening negation

If we have a negation with wide scope, like in (279a) and represented it as (279b), then by applying DeMorgan, we get a disjunction, as in (279c). At least one of the original conjuncts needs to be false, for the sentence to be true. At first glance this seems correct (either it isn't every man, or it isn't a book or it isn't reading or it isn't happening at present...) At least one of the truth conditions of the material in the scope of the negation is not fulfilled. As we have seen in section 3.1.2, a wide scope reading for negation is the most general one. In focus readings, the negation would then pick out a specific conjunct.

(279) a. Het is niet zo dat elke man een boek leest.
it is not so that every man a book reads
'It is not the case that every man reads a book.'

b. not.[man(X+decr+every+[]) &
book(Y+incr+some+[X]) &
read(E+incr+some+[X]) &
event(E+incr+some+[X]) &
agent_of(E+incr+some+[X],X+incr+every+[]) &
theme_of(E+incr+some+[X],Y+incr+some+[X]) &
attime(E+incr+some+[X],T) &
tense(E+incr+some+[X],pres)
]

c. not.man(X+decr+every+[]) ;
not.book(Y+incr+some+[X]) ;
not.read(E+incr+some+[X]) ;
not.event(E+incr+some+[X]) ;
not.agent_of(E+incr+some+[X],X+incr+every+[]) ;
not.theme_of(E+incr+some+[X],Y+incr+some+[X]) ;
not.attime(E+incr+some+[X],T) ;
not.tense(E+incr+some+[X],pres)

There are however serious problems with this way of treating negation. First of all, it does not make sense to negate all conjuncts independently. Negation scopes over a quantifier or it doesn't. If it does, it affects all occurrences of the quantifier in the representation. Second, when negation scopes over a quantifier it changes its entailment directions, as we have seen in section 4.4.2. Negations as structural operators can hardly be integrated in the approach developed there. The entailment directions were changed under the influence of governors in the fourth slot of the variable's quadruple. This is a

flattened way of treating higher operators. Such a flattening is also needed for negation to make it fit into the system. Third, for maximum flatness, the conjunction should ideally be the highest operator.

The simplest solution is to let negation always target a quantifier, namely the highest one that it has scope over. That way we get rid of 'loose' negations. FLF then does not contain logical negation, just like it does not contain logical quantifiers. In that sense it is structurally comparable to description logics.

The disadvantage of this solution is that it complicates the application procedure that derives FLF. Negation must attach to a quantifier and change its entailment properties. These changed properties must then be used in the dependency relations which influence the entailment directions of dependent quantifiers. This is inconsistent with the non-nesting policy explained in section 4.4.2.

An alternative solution is to let the negation form a conjunct of its own in FLF. Quantifiers can then be marked as being dependent on it. Dependence on a negation has the same effect as dependence on a decreasing quantifier. It is, however, not clear how exactly the negation conjunct is best represented; does it introduce a predicate (if so, which) and a variable, is it relevant to mark what it is dependent on, etc.? Also, the status of such a conjunct is problematic, because it is not clause-like, like the other conjuncts. We take the position that the negative conjunct itself can be dispensed with, but that its scope should be marked, as if it were there, on other conjuncts in their fourth slot. (279a) is then represented as (280c) with (280b) as an intermediate step.

(280) a. Het is niet zo dat elke man een boek leest.
it is not so that every man a book reads
'It is not the case that every man reads a book.'

b. man(X+decr+every+[neg+decr]) &
book(Y+incr+some+[X,neg+decr]) &
read(E+incr+some+[X,neg+decr]) &
event(E+incr+some+[X,neg+decr]) &
agent_of(E+incr+some+[X,neg+decr],X+incr+every+[neg+decr]) &
theme_of(E+incr+some+[X,neg+decr],Y+incr+some+[X,neg+decr])&
attime(E+incr+some+[X,neg+decr],T) &
tense(E+incr+some+[X,neg+decr],pres)

c. man(X+incr+every+[neg]) &
book(Y+decr+some+[X,neg]) &
read(E+decr+some+[X,neg]) &
event(E+decr+some+[X,neg]) &
agent_of(E+decr+some+[X,neg],X+decr+every+[neg]) &
theme_of(E+decr+some+[X,neg],Y+decr+some+[X,neg]) &
attime(E+decr+some+[X,neg],T) &
tense(E+decr+some+[X,neg],pres)

As there may be more than one negation in a sentence, *neg*s need to be numbered. We will see an example of this in the next section.

This is the most minimal way to do it. One may ask whether it is possible to treat negation in a way that is parallel to modal adverbs, via the evidence for a proposition (see section 4.3). The evidence would then be negative. The problem is that this offers no straightforward way to transfer the decreasing entailment property, since the proposition is assumed to have a definite quantifier because of its transparency. Therefore the variable of the proposition cannot carry this property and the *evidence* predicate does not introduce an extra variable. Moreover, negation is not an intensional operator. Only intensional operators should introduce an intensional domain, such as *proposition*, as this extra layer stands for intensional embedding.

Negation is therefore represented by marking its scope on the conjuncts that are in its scope. Negations bring in downward monotonicity.

4.5.2 DeMorgan

With this approach we can formulate the DeMorgan rules for quantifiers at FLF. For (246), repeated here as (281), (282b) and (282c) follow from (282a) by the DeMorgan rules for quantifiers.

(281) Geen man heeft elk boek gelezen.
no man has every boek read
'No man has read every book.'

(282) a. $\neg\exists x[man^{\downarrow}(x)\&\forall y[book^{\uparrow}(y) \rightarrow \exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$
geen man heeft elk boek gelezen
no man has read every book

b. $\forall x[man^{\downarrow}(x) \rightarrow \neg\forall y[book^{\uparrow}(y) \rightarrow \exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$
elke man heeft niet elk boek gelezen
every man has not read every book

c. $\forall x[man^{\downarrow}(x) \rightarrow \exists y[book^{\uparrow}(y)\&\neg\exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$
voor elke man is er een boek dat hij niet heeft gelezen
for every man there is a book that he didn't read

These effects can be reproduced in FLF as illustrated in (283). The first step shown is just negation influencing the entailment directions. Then, one quantifier per step is moved out of the scope of negation, starting with the highest one. This means that all dependencies need to be marked in this stage, because it needs to be known which is the highest quantifier. Dependencies that pose no further conditions on entailment can possibly be deleted afterwards. Another way to increase efficiency is to mark the nature of the dependencies from the beginning, so that it is clear which ones affect referentiality and which don't. The entailment directions, unsurprisingly, stay as they are. Thus, (283b) corresponds to (282a), (283c) to (282b), and (283d) to (282c). These are three equivalent forms.

(283) a. man(X+incr+some+[neg+decr]) &
book(Y+decr+every+[neg+decr]) &
read(E+incr+some+[Y,neg+decr]) &
event(E+incr+some+[Y,neg+decr]) &
agent_of(E+incr+some+[Y,neg+decr],X+incr+some+[neg+decr]) &
theme_of(E+incr+some+[Y,neg+decr],Y+incr+every+[neg+decr]) &
attime(E+incr+some+[Y,neg+decr],T)

b. man(X+decr+some+[neg]) &
book(Y+incr+every+[X,neg]) &
read(E+decr+some+[Y,X,neg]) &
event(E+decr+some+[Y,X,neg]) &
agent_of(E+decr+some+[Y,X,neg],X+decr+some+[neg]) &
theme_of(E+decr+some+[Y,X,neg],Y+decr+every+[X,neg]) &
attime(E+decr+some+[Y,X,neg],T)

c. man(X+decr+**every**+[]) &
book(Y+incr+every+[neg]) &
read(E+decr+some+[Y,X,neg]) &
event(E+decr+some+[Y,X,neg]) &
agent_of(E+decr+some+[Y,X,neg],X+decr+**every**+[]) &
theme_of(E+decr+some+[Y,X,neg],Y+decr+every+[X,neg]) &
attime(E+decr+some+[Y,X,neg],T)

d. man(X+decr+every+[]) &
book(Y+incr+**some**+[]) &
read(E+decr+some+[Y,X,neg]) &
event(E+decr+some+[Y,X,neg]) &
agent_of(E+decr+some+[Y,X,neg],X+decr+every+[]) &
theme_of(E+decr+some+[Y,X,neg],Y+decr+**some**+[X]) &
attime(E+decr+some+[Y,X,neg],T)

The function of the boldface text is to help the reader spot the differences with respect to the previous step. (Of course all occurrences of a quantifier need to be adapted. They are identified by the variable that they are marked on.) For a more unified approach the quantifier *geen* 'no' has been split up into a negation operator plus a quantifier *some*. Additional motivation for this comes from split scope constructions (Penka and Zeijlstra, 2005), which we discussed in section 2.3.4. Another advantage is that dependence on negation is uniformly marked locally. The steps can in principle also be applied in the inverse order, moving the quantifiers one by one under the scope of negation. In this example there was only one negation. When a sentence contains more than one negation, like (284), an extra rule is needed that cancels two negations against each other.

(284) Niemand werkt niet.
no-one works not
'No-one doesn't work.' (double negation)

In the FLF of this sentence, (285), I have added neg_1 and neg_2 between the conjuncts as an extra indication of the scope of the negations. Note that this is purely for the purpose

of visualization. The order of the conjuncts is irrelevant and scope is indicated by the dependencies.

(285) *FLF of (284)*
neg$_1$
person(X+decr+some+[neg$_1$]) &
neg$_2$
work(E+incr+some+[neg$_2$,X,neg$_1$]) &
event(E+incr+some+[neg$_2$,X,neg$_1$]) &
agent_of(E+incr+some+[neg$_2$,X,neg$_1$],X+decr+some+[neg$_1$]) &
attime(E+incr+some+[neg$_2$,X,neg$_1$],T)

If we now move the highest quantifier out of the scope of negation by DeMorgan, as above, we get (286), in which nothing intervenes between the scope of the two negations. (In fact, the representation does not even tell which of the two is the higher one.)

(286) *from (285) by DeMorgan for quantifiers*
person(X+decr+every+[]) &
neg$_1$
neg$_2$
work(E+incr+some+[neg$_2$,X,neg$_1$]) &
event(E+incr+some+[neg$_2$,X,neg$_1$]) &
agent_of(E+incr+some+[neg$_2$,X,neg$_1$],X+decr+every+[]) &
attime(E+incr+some+[neg$_2$,X,neg$_1$],T)

These two negations can be canceled against each other, by a rule of negation elimination, yielding (287),which is identical to the representation of (288).

(287) *from (286) by negation elimination*
person(X+decr+every+[]) &
work(E+incr+some+[X]) &
event(E+incr+some+[X]) &
agent_of(E+incr+some+[X],X+decr+every+[]) &
attime(E+incr+some+[X],T)

(288) Iedereen werkt.
everyone works
'Everyone works.'

Doing this the other way round is unpractical from a procedural perspective, because you can keep adding negations, i.e. the process has no natural end. This rule should therefore only be applied in one direction. In other words, if the text contains a sentence meaning P, it is not very sensible to produce an equivalent form ¬¬P, to match a possible hypothesis ¬¬P, even though this is a valid equivalent. It is much better to produce an equivalent form P for a hypothesis of the form ¬¬P, if it occurs.

For applying DeMorgan at FLF, we thus need two rules: one to move quantifiers out of the scope of the negation, replacing them by their dual, their DeMorgan counterpart,

so to say, (see Barwise and Cooper (1981) for a mathematical definition of dual), and one to cancel two negations against each other, when nothing scopes between them.

The implementation of DeMorgan in FLF requires a list of quantifiers and their duals (e.g *every* changes to *some* and vice versa). Homomorphisms (proper names and singular definites) are selfdual, that is, they do not change when moved out of the scope of a negation. For many other quantifiers the DeMorgan rules do not apply.

An important question with respect to the representation of texts is whether to keep all DeMorgan variants of a sentence available. As all the DeMorgan variants of a sentence are logically equivalent, deriving a normal form and using only that suffices for purposes of logical inference. Discourse related processes, such a anaphora resolution, need the original representation of the sentence. It is best to assign the variant in which the negation is lowest, the status of normal form. The DeMorgan rules must then also be applied to the query, to bring it in normal form.

4.5.3 Splitting decreasing quantifiers

Splitting up *geen* 'no' into negation plus *some* prompts the question whether certain other decreasing quantifiers such as *weinig* 'few'[6] should also be analyzed with a separable negation. At least two diagnostics to decide on this issue present themselves. One could test whether the quantifier can give rise to split scope readings and whether it supports DeMorgan-like equivalence patterns. That is, is there a quantifier Q for which (289) holds?

(289) Weinig mannen slapen ⇔ Q mannen slapen niet
few men sleep Q men sleep not
Few men are asleep ⇔ Q men are not asleep

Veel 'many' seems to present itself as a candidate for Q (which would require many to be its own DeMorgan counterpart, on the analysis of *few* as *not many*), but this is problematic. It only holds when the number of men who sleep is considered small in comparison to the total set of men, but not when it is considered small with respect to the total set of sleepers (or to some other subset of sleepers, e.g. women who sleep). It thus fails the DeMorgan test. That means that, at least for the correct functioning of the DeMorgan rules, it is not necessary to be able to split it.

For split scope effects on the other hand, it does appear to be necessary to also split up determiners like *weinig*. (290) seems to have all three readings, parallel to (291) (repeated from chapter one), though some judge the (b) reading as a bit odd.

(290) Ze mogen weinig verpleegkundigen ontslaan.
they may no nurse fire

a.'They are not allowed to fire many nurses. $\neg > may > many$
b.(?)'There are few nurses who they are allowed to fire.' $\neg > many > may$
c.'They are allowed not to fire many nurses.' $may > \neg > many$

[6] *Weinig* is actually not neatly decreasing, but it patterns in some ways with decreasing quantifiers.

(291) Ze mogen geen verpleegkundige ontslaan.
they may no nurse fire

a. 'They are not allowed to fire any nurse. $\neg > may > \exists$
b. 'There is no nurse who they are allowed to fire.' $\neg > \exists > may$
c. 'They are allowed not to fire a nurse.' $may > \neg > \exists$

This observation is strengthened if we use the modal verb *hoeven*, which is a negative polarity item (NPI). Here the (c) reading is ruled out, because the NPI is not licensed, since with *weinig* completely scoping below it, *hoeven* is not in a decreasing environment.[7] As the (b) reading is still found a bit odd, the (a) reading, which is the split reading that we are after, is clearly the most likely reading.

(292) We hoefden weinig boeken te lezen.
we needed few books to read

a. 'We were not required to read many books. $\neg > may > many$
b.(?)'There were not many books we were required to read.' $\neg > many > may$
c.*'We were required not to read many books.' $may > \neg > many$

What exactly are the composing parts of such a decreasing quantifier is however a puzzling question. For *geen* the situation seems clear. If it decomposes into antimorphic negation and an additive existential quantifier, it is to be expected that *geen* itself, the composition of these two elements, is anti-additive, which is indeed the case.

Since the raising behavior that the negative part shows is rather peculiar, we would expect the raising element to be the same in all cases. Therefore the default assumption would be that the negative part that is split off in the case of *weinig*, is also the normal antimorphic negation. Whether the negative part that is split off is indeed a normal negation or something weaker, is difficult to test. Testing the strength of the separated negation would require an NPI modal verb that can only be licensed by strong negation. At least in Dutch, such a modal verb has not been reported to exist. Let us thus assume that one of the two parts that make up *weinig* (and other decreasing quantifiers) is a regular negation. Then we still need to figure out what the other part is. It would be tempting to think that this is its increasing counterpart *veel* 'many'. Zwarts (1981) however, argues against analyzing *weinig* as *niet veel*. On the one hand he points out that there is a danger of circularity, since one could just as well analyze *veel* as *niet weinig*. Here the split scope effect would provide independent evidence to favor the former, since

[7] Notice in this context also that, as Ton van der Wouden pointed out to me, decreasing quantifiers seem to be ungrammatical in the complement of positively polar modal verbs. *moeten* is weakly positive polar and *dienen* is a real positive polarity item (PPI).

(1) a. We hoeven maar drie roeiers uit te nodigen.
b. ? We moeten maar drie roeiers uitnodigen.
c. * We dienen maar drie roeiers uit te nodigen.

veel does not show split scope effects. But Zwarts also observes that (293) is not a contradiction, which it should be if *weinig* meant the same as *niet veel*[8].

(293) Er zijn er niet veel, maar ook niet weinig.
there are there not many but also not few
'They aren't many, but there aren't few either.'

Lappin (2000) gives an intensional parametric account for the interpretation of *many* and *few*, defined in (294a) and (294b) respectively, where *sa* is the actual situation and *S* is a set of normative situations *sn*. A possible situation *sn* is normative in that it provides a case in which the number of objects in the intersection of the A an B sets is large enough to provide a standard for comparison for the assertion that many A are B in the actual situation. Different extensional readings can be derived from the intensional ones by placing additional constraints on *S*.

(294) a. $[\![\text{B}]\!]^{sa} \in [\![\text{many}]\!]([\![\text{A}]\!]^{sa})$ iff $\text{S} \neq \emptyset$, and for any $\text{sn} \in \text{S}$,
$|[\![\text{A}]\!]^{sa} \cap [\![\text{B}]\!]^{sa}| \geq |[\![\text{A}]\!]^{sn} \cap [\![\text{B}]\!]^{sn}|$

b. $[\![\text{B}]\!]^{sa} \in [\![\text{few}]\!]([\![\text{A}]\!]^{sa})$ iff $\text{S} \neq \emptyset$, and for any $\text{sn} \in \text{S}$,
$|[\![\text{A}]\!]^{sa} \cap [\![\text{B}]\!]^{sa}| < |[\![\text{A}]\!]^{sn} \cap [\![\text{B}]\!]^{sn}|$

Few and *many* are not actually monotone because the set of normative situations normally does not stay the same, when one of the argument sets is replaced with a superset or subset. For example, if many men read a regional novel, those men (and possibly some others) read a book. But typically, that same number no longer counts as many, when it is the number of men reading a book. The upward monotonicity gets into trouble, because with going to a superset (less specific) the standard also tends to go up and might therefore not be met. For *few*, downward monotonicity is likewise damaged by the standard going down. Yet they go in opposite directions, as *many* is evaluated with respect to a minimum and *few* is with respect to a maximum. If we take *not many* to be the complement of many, the definition of *not many* is exactly the same as that of *few*.

(295) $[\![\text{B}]\!]^{sa} \in [\![\text{not}]\!]([\![\text{many}]\!]([\![\text{A}]\!]^{sa}))$ iff $\text{S} \neq \emptyset$, and for any $\text{sn} \in \text{S}$,
$|[\![\text{A}]\!]^{sa} \cap [\![\text{B}]\!]^{sa}| < |[\![\text{A}]\!]^{sn} \cap [\![\text{B}]\!]^{sn}|$

Normally, however, different standards for comparison are used for *many* and *few*. The maximum for *few* is not usually the same as the minimum for *many*. Therefore, there may be certain values for the cardinality of the intersection of A and B, that neither counts as

[8]This may be due to the vagueness that is more generally observed in predication. *Few* and *many* have a partly predicative character, rather than being purely quantificational. Like (1a), (1b) is with the right intonation, where the second *niet* forms a unit with *veel*, not necessarily felt as contradictory

(1) a. Het is niet groot, maar ook niet niet groot.
it is not big but also not not big
'It is not big, but it isn't not big either.'

b. Er zijn er niet veel, maar ook niet niet veel.
there are there not many but also not not many
'They aren't many, but there aren't not many either.'

few, nor as *many*. There typically is such a grey area of a more or less normal amount, just like with adjective pairs like *tall* and *short*. This explains why sentences like (293) are not contradictory. When *few* is split up into a negation and another part, this other part is like *many*, but with the parameter settings of *few*, let's call it *many*$^+$, as it includes a wider range of cases than *many*.

many$^+$ does not occur as a separate determiner, but only as part of *few*. In this light it is interesting that Seuren (2006) argued that *no* is different from *not some*, or actually, that the *some* in *not some*, and by itself, is different from the 'some' in *no*. In his approach the basic natural meaning of *some* excludes *all* (*some A are B* entails *not all A are B*). He calls this 'exclusive some'. *No* on the other hand is the negation of the 'inclusive some', that corresponds to the standard logical existential quantifier.

Numerical quantifiers of the type *at most n*, on the other hand, neatly split up in negation plus *more than n*.

(296) Ze hoeven hoogstens drie verpleegkundigen te ontslaan.
they need at most three nurses to fire

a.'It is not necessary for them to fire more than three nurses. $\neg > \Box > morethan3$
b.(?)'There are at most three nurses who they need to fire.' $\neg > morethan3 > \Box$

Psycholinguistic experiments by Geurts and van der Slik (2005a) provide additional evidence that decreasing quantifiers are more complex than increasing quantifiers. Subjects take longer and make more mistakes in reasoning with decreasing quantifiers. The authors suggest that decreasing and increasing quantifiers may exhibit the same kind of asymmetry as adjectives associated with opposite ends of the same scale. There, the one associated with the upper end of the scale is the simpler and more neutral one and often considered basic, whereas the other one is derived from it.

It goes too far, at this point, to claim that all decreasing quantifiers contain a negation. Universal quantifiers are decreasing on their left argument and it is not clear how they should be analyzed along these lines. Yet, for the vast majority of decreasing quantifiers, I conclude that it is reasonable to analyze them as containing a separable negation.

4.5.4 Consistent and complete entailment patterns

The definitions of consistent and complete NPs are repeated here.

(297) a. consistent: $f(-X) \subseteq -f(X)$
NP (do) not VP $\Rightarrow$ It is not the case that NP VP
b. complete: $-f(X) \subseteq f(-X)$
It is not the case that NP VP $\Rightarrow$ NP (do) not VP

Generalized quantifiers that are both consistent and complete (homomorphisms) are selfdual (their own DeMorgan counterpart, underspecification and disjunction), as observed in section 4.5.2. I am not aware of the existence of any that are complete, but not consistent. However, Kas (1993) does show that there are some which are consistent

but not complete. The examples he gives are *both N*, *the n N* and *the N(pl)*. Consider (298a) and (298b). That *the three men* is consistent, is shown by the fact (298a) entails (298b). That *the three men* is not complete is shown by the fact that (298b) does not entail (298a). Suppose that two of the three men work, then (298b) is true, but (298a) is not.

(298) a. The three men do not work.
b. It is not the case that the three men work.

The rule for consistent quantifiers is like the DeMorgan rule for homomorphisms, except that it only holds in one direction. Instead of producing an equivalent formula, it only produces an entailment. As long as we only move the negation over one quantifier at a time and then check if a next step is possible, we do not deal with composed functions, and hence do not need a calculus to compute the effects.

4.5.5 Summary

Negation is only represented in the fourth slot of a variable's quadruple of the conjuncts that are in its scope. It comes with a decreasing property, which takes effect in the way we have seen in the previous section. Most decreasing quantifiers can be analyzed as containing a separable negation. DeMorgan rules for quantifiers are applied to derive a normal form where the negation is as low as possible.

4.6 Underspecification and disjunction

In case one QLF in its 'spell out' gives rise to real ambiguity, this ambiguity is expressed in terms of scopal dependencies. This section explores whether, if ambiguity occurs, FLF can take the form of a single conjunction of disjunctions, representing the complete class of readings. A conjunct would then internally be a disjunction iff a variable occurring in it may or may not be referentially dependent.

Using disjunction to represent different readings is notoriously problematic because of its interaction with negation. The DeMorgan rule for negation over disjunctions ($\neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$), would turn disjunctions into conjunctions, negating all possible readings of a sentence at the same time. If negation is completely excluded from the (meta level) logic we use for our representations, this opens up possibilities for underspecification of scope ambiguities by means of disjunction.

When a generalized quantifier can be referentially either dependent or independent, all predicates in which its variable is bound have two options: one in which the fourth position of the variable's quadruple contains a governor and one in which it is empty. This could be written as a disjunction as illustrated in (299). Note that this is the result of the merging of two fully specified analyses.

(299) a. Elke man leest een boek.
every man reads a book
'Every man reads a book.'

b. man(X+decr+every+[]) &
book(Y+incr+some+[X]) ; book(Y+incr+some+[]) &
read(E+incr+some+[X]) &
event(E+incr+some+[X]) &
agent_of(E+incr+some+[X],X+incr+every+[]) &
theme_of(E+incr+some+[X],Y+incr+some+[X]) ;
theme_of(E+incr+some+[X],Y+incr+some+[]) &
attime(E+incr+some+[X],T) &
tense(E+incr+some+[X],pres)

Of course, if the disjunction concerning one occurrence is resolved, it is also resolved for the other occurrences in the same way. A reading without disjunctions in which the same variable in one place is marked as dependent and in another place as independent is not possible. This is actually better represented in (300)

(300) . . . &
(book(Y+incr+some+[X]) &
theme_of(E+incr+some+[X],Y+incr+some+[X]));
(book(Y+incr+some+[]) &
theme_of(E+incr+some+[X],Y+incr+some+[])) &
. . .

Similarly, it should be prevented that quantifiers can be dependent on each other. *Many*, for example can be dependent on other quantifiers, but other quantifiers can also be dependent on *many*. When the different readings of a sentence with two or more occurrences of *many* are merged into one, it should be prevented that it is an option that each of the two is dependent on the other. This can also be done with extra bracketing like above.

(301) (pred1(X+nonm+many+[] & pred2(Y+nonm+many+[X]) ;
(pred1(X+nonm+many+[X] & pred2(Y+nonm+many+[])

If there is a third reading where both are independent, this option will need to be added as an extra disjunct (or within both existing ones as a extra option for the one that is now dependent). Similar problems can be expected to occur in all cases where the choice in one disjunction has consequences for the choices in others. One always needs to beware of ruled out readings sneaking back in.

A solution to problems of this type is to mark each conjunct for the sentence and reading(s) it belongs to. Something like this will anyway be necessary for processing text, because a sequence of different sentences needs to be treated differently from different readings for the same sentence. Material that occurs in all readings can be inferred independently of the choice of the reading. This way of marking sentences and readings is illustrated in (302), a revised representation of (299), in the format CONJUNCT:sentence_number:[reading$_1$,. . . reading$_n$] ((299) is here represented as the first sentence of a text).

(302) man(X+decr+every+[]):1:[1,2] &
book(Y+incr+some+[X])1:[1] ; book(Y+incr+some+[]):1:[2] &

read(E+incr+some+[X]):1:[1,2] &
event(E+incr+some+[X]):1:[1,2] &
agent_of(E+incr+some+[X],X+incr+every+[]):1:[1,2] &
theme_of(E+incr+some+[X],Y+incr+some+[X]):1:[1] ;
theme_of(E+incr+some+[X],Y+incr+some+[]):1:[2] &
attime(E+incr+some+[X],T):1:[1,2] &
tense(E+incr+some+[X],pres):1:[1,2]

This way of representing ambiguity can be considered an abbreviated notation. In the entailment process it is not allowed to combine information that does not occur together in one reading. The written conjunction is then a reflection of a higher disjunction, an exclusive disjunction between complete readings, exclusive, not in the sense of truth and falsity, but in the sense that only one reading at the time may be consulted for entailment. Looking into one, makes the others unavailable. It is not essential to write a disjunction, a conjunction will do just as well. The 'disjunctivity' is sufficiently encoded in the indexes and the way they are treated. The different readings are merged together into one representation, but can always be reconstructed as separate readings. The representation of (299) with bookkeeping of sentence and readings is then (303).

(303) man(X+decr+every+[]):1:[1,2] &
book(Y+incr+some+[X])1:[1] &
book(Y+incr+some+[]):1:[2] &
read(E+incr+some+[X]):1:[1,2] &
event(E+incr+some+[X]):1:[1,2] &
agent_of(E+incr+some+[X],X+incr+every+[]):1:[1,2] &
theme_of(E+incr+some+[X],Y+incr+some+[X])1:[1] &
theme_of(E+incr+some+[X],Y+incr+some+[]):1:[2] &
attime(E+incr+some+[X],T):1:[1,2] &
tense(E+incr+some+[X],pres):1:[1,2]

The disjunction can better be reserved for linguistic disjunction. I assume with Pietroski (2006) and Jaspers (2005) that conjunction is the basic operation in language. As Pietroski puts it:

> "each complex expression of natural language is the concatenation of two simpler expressions; these two constituents, together with the meaning of concatenation, determine the meaning of the complex expression; constituents are understood as monadic *predicates* and concatenation signifies *conjunction*. so from a semantic perspective every complex expression is a conjunction of predicates."

This intuitively makes sense. With everything you say, you add information. Disjunction is an exceptional case. It complicates reasoning considerably. If you know *A and B*, you know that both *A* and *B* are the case. If you know *A or B* then you don't know whether *A* is the case, nor whether *B* is the case. Introducing disjunction that can interact with the conjunctions in FLF is most likely unavoidable, and the exploitation of disjunctions for entailment can be expected to require extra reasoning steps.

4.7 Entailment on FLF

The main purpose of the flat representation is inferential: to decide whether a certain inference is possible, it suffices to linearly inspect an FLF and, for each conjunct, to decide locally whether or not it gives rise to (part of) the hypothesis induced by the query.

Quantifiers are represented in a 'flattened' way. They are removed from the logic in which the representation is cast, but the information about them is retained. The representation is, on the surface, a conjunction, so the only thing that follows from it by logic is the entailment of each conjunct. From the quantificational information nothing follows immediately, but the information is available to rules (comparable to meaning postulates) that can derive additional entailments. Because of this, even natural language quantifiers of which the translation to first-order logic is problematic, can be handled. Each variable is bound in its propositional domain.

4.7.1 Basic conjunctive entailment

Entailment on FLF applies the elementary insight that a conjunction of closed propositions (propositions without free variables) entails each of its conjuncts. In (304) a recursive definition of this concept of entailment is given.

(304) conjunctive entailment

0. p entails p
1. a conjunction of propositions p_1 & p_2 & .... & p_n entails p_i for every *i* between 1 and *n* ($1 \leq i \leq n$)
2. p_1 & p_2 & .... & p_n entails p_i & p_j iff the conjunction entails both p_i and p_j
3. p_1 & p_2 & .... & p_n entails p_r if there is an explicit inferential relation stated between p_i and p_r and the conjunction entails p_i

Statement (2) generalizes (1). Statement (3) sees to the situation that additional specifications from ontologies, structured thesauri or extended lexicons are applied to a certain corpus of texts, providing additional inferences for individual propositions. The application of statement (3) is mediated by the entailment properties:

- P(Var+incr+Quant+Dep) entails P↑(Var+incr+Quant+Dep), where P↑ denotes a superset of P
- P(Var+decr+Quant+Dep) entails P↓(Var+decr+Quant+Dep), where P↓ denotes a subset of P
- P(Var+Dir+Quant+Dep) entails Q(Var+Dir+Quant+Dep), when P and Q denote the same set

This is the basic idea for conjunctive entailment, but it does not suffice entirely.

4.7.2 Modifications for conjunctions in non-increasing contexts

Conjunctive entailment as defined above, works well in increasing contexts. In decreasing or non-monotone contexts modifications are needed.

When two or more propositions containing the same variable are marked as decreasing, they cannot be entailed independently, but only in combination. The entailment in (305a) does not hold, even though on the basis of statement (2) above, the premise, as represented in (305b), should entail the conclusion, as represented in (305c). (As ambiguity is not relevant for the discussion here, I omit the bookkeeping of sentences and readings.)

(305) a. elke blonde man werkt $\not\Rightarrow$ elke man werkt
every blond man works $\not\Rightarrow$ every man works

b. *elke blonde man werkt*
man(A+decr+every+[]) &
blond(A+decr+every+[]) &
work(B+incr+some+[A]) &
event(B+incr+some+[A]) &
agent_of(B+incr+some+[A],A+incr+every+[]) &
attime(B+incr+some+[A],C) &
tense(B+incr+some+[A],pres)

c. *elke man werkt*
man(A+decr+every+[]) &
work(B+incr+some+[A]) &
event(B+incr+some+[A]) &
agent_of(B+incr+some+[A],A+incr+every+[]) &
attime(B+incr+some+[A],C) &
tense(B+incr+some+[A],pres)

A solution is to sort the conjuncts first by variable and then by entailment property. For each variable A we then get $\mathcal{P}_A$, the set of all conjuncts in which A is bound, which is then divided into $\mathcal{P}_A{}^-$, the set where A is marked as decreasing, $\mathcal{P}_A{}^+$, the set where A is marked as increasing, and $\mathcal{P}_A{}^0$, the set where A is marked as non-monotone.

For our premiss above, the sorted version is given in (306). (Some conjuncts now occur more than once, because they contain more than one bound variable, but this does not matter.) The sorting can be done off-line.

(306) $\mathcal{P}_A{}^-$ = {P_A| A bound in P and decreasing}:
man(A+decr+every+[]) &
blond(A+decr+every+[])
$\mathcal{P}_A{}^+$ = {P_A| A bound in P and increasing}:
agent_of(B+incr+some+[A],A+incr+every+[])
$\mathcal{P}_B{}^+$:
work(B+incr+some+[A]) &
event(B+incr+some+[A]) &
agent_of(B+incr+some+[A],A+incr+every+[]) &

attime(B+incr+some+[A],C) &
tense(B+incr+some+[A],pres)
$\mathcal{P}_C{}^+$:
attime(B+incr+some+[A],C)[9]

Subsets are entailed of the intersection of all predicates on which A's quantifier is decreasing. That is, all entailed sets lay within the intersection. A subset of ⟦man⟧∩ ⟦blond⟧is, for example, ⟦man⟧∩ ⟦blond⟧∩ ⟦young⟧, but not, for example, ⟦man⟧(that would be a superset). For the predicates on which A's quantifier is increasing, on the other hand, supersets of the intersection are entailed.

On the other side, if several predicates over the same variable in a decreasing environment occur in the query, only one of them needs to be found in the text.

As an alternative to off-line preparation, as sketched above, it is also possible to build the adjustment into the inference procedure entirely. Suppose (305c) is a hypothesis for which we want to check entailment by some text. First we look at the first conjunct *man(A+decr+every+[])*. It is marked as decreasing. We check if there are more decreasing conjuncts with the same variable. This is not the case. Now we try to find a match for this conjunct in the text. Suppose we find a matching conjunct in (305b). Because this is a decreasing conjunct, we need to check if there are more decreasing conjuncts with the same variable in the sentence. If this is not the case we have found a good match for the first conjunct. In (305b), however, there is another decreasing conjunct with the same variable, namely *blond(A+decr+every+[])*. Since no such conjunct occurred in the hypothesis, as we already verified in the beginning, it is concluded that (305b) cannot entail (305c). Let us now consider a scenario where (305b) is the hypothesis to be tested. The first conjunct is again *man(A+decr+every+[])*. But this time there is another decreasing conjunct in the hypothesis with the same variable *blond(A+decr+every+[])*. Now if we find (305c) in the text, we see that it contains *man(A+decr+every+[])* and no other decreasing conjuncts with the same variable. This conjunct now suffices as a good match for the first two conjuncts of the hypothesis. Such a procedure seems feasible and not necessarily more complicated than checking for dependencies. The two alternatives have to be compared and evaluated in implementation.

This problem arose, because the conjunctions between predicates under a decreasing quantifier should actually be interpreted as being in a decreasing environment, too, but they are not marked as such in the flat representation. Compare this to a first-order logic representation. In a traditional representation of *elke blonde man werkt* (∀x. man(x) ∧ blond(x) → work(x)), the conjunction between *man(x)* and *blond(x)* is also in a decreasing environment. Here this is not explicitely marked, but the corresponding behaviour in terms of entailment, follows from the rules of the logic. The solution proposed for FLF, in either implementation, boils down to grouping the predicates marked as decreasing, in order to specify which conjunctions are to be interpreted in a decreasing context. An alternative would be to work with brackets and/or markers on the conjunction symbols themselves to build in a more principled grouping of predicates

[9] *C* is taken to be a referring constant (remember tense was treated as a pronoun), and therefore increasing and insensitive to negation.

in decreasing environments in the process of deriving FLF. It is at present not clear if this is possible.

Note that trying to extract the conjunction from its decreasing environment by turning it into a disjunction only would work for a very limited set of cases. The pair in (307), where *and* is in an antimorphic environment, is indeed equivalent. With anti-additive *nobody*, the equivalence already does not hold anymore. Though (308b) entails (308a), (308a) does not entail (308b). This is the general pattern for monotone decreasing quantifiers, as Zwarts (1981) already observed.

(307) a. Alice does not dance and sing.
b. Alice does not dance or Alice does not sing.

(308) a. Nobody dances and sings.
b. Nobody dances or nobody sings.

In non-monotone contexts, several predicates over the same variable always need an exact match. There may not be any in the text that are not in the hypothesis, and there may not be any in the hypothesis that are not in the text.

4.7.3 Entailments between quantifiers

The logic that concerns entailments between quantifiers, takes the form of a set of rules. It is not necessary to stick with the standard modern predicate calculus. Seuren (ming) shows that there are more intuitive alternatives, which, for instance, preserve the Aristotelian entailments that were rejected in modern logic. He hypothesizes that humans have an innate basic natural logic, which can be upgraded to strict natural logic. Standard modern logic as developed by mathematicians is only available through specialistic education on the topic.

Still, the logics that Seuren discusses are so far limited to *some*, *all* and, in some variants, *no*. We also can include entailment rules for non-standard quantifiers. Many of these have at least existential entailments. Examples are *more than half of the N*, *most N*, *at least n N*, *all but n N*. These all give rise to entailments in which they are replaced with *some N*.

Also the universal quantifier *all* intuitively has an existential entailment (I will indicate this entailment rule with $\forall \Rightarrow \exists$), as demonstrated by Aristotelian logic. This follows, if we assume, inspired on Seuren (2006) that universal quantification in natural language is not to be modeled with empty restrictors. As an example, we show the kind of rule that gives us such entailments. When a universal quantifier is the highest operator in a sentence you may infer a sentence where this universal quantification is substituted with an existential one. (On the claim that if you are talking about 'all men', you are saying that there are men in your domain, which in turn means that there are men in the domain for which the predicate holds that holds for all men.) The new existential quantifier gives a normal upward entailment on *man*, the other entailment directions in the sentence remain as they were. This is illustrated in (310) for the DeMorgan equivalents of (246), repeated in (309.

(309) a. $\neg\exists x[man^{\downarrow}(x)\&\forall y[book^{\uparrow}(y) \rightarrow \exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$
geen man heeft elk boek gelezen
no man has read every book

b. $\forall x[man^{\downarrow}(x) \rightarrow \neg\forall y[book^{\uparrow}(y) \rightarrow \exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$
elke man heeft niet elk boek gelezen
every man has not read every book

c. $\forall x[man^{\downarrow}(x) \rightarrow \exists y[book^{\uparrow}(y)\&\neg\exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$
voor elke man is er een boek dat hij niet heeft gelezen
for every man there is a book that he didn't read

(310) a. from (309b), by $\forall \Rightarrow \exists$:
$\exists x[man^{\uparrow}(x)\&\neg\forall y[book^{\uparrow}(y) \rightarrow \exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$

b. from (309c), by $\forall \Rightarrow \exists$:
$\exists x[man^{\uparrow}(x)\&\exists y[book^{\uparrow}(y)\&\neg\exists e[read^{\downarrow}(e)\&ag(e,x)\&th(e,y)]]]$

The $\forall \Rightarrow \exists$ rule may only be applied to the original formula and its DeMorgan equivalents, not to its logical consequences that are derived by making use of the entailment properties. So, from our original sentence follows (311a), by downward entailment on man, and from that, by DeMorgan, follows (311b). But it is at least questionable whether it is allowed to apply the $\forall \Rightarrow \exists$ rule here to derive (311c), because then the original sentence would have had to claim that there are clever men in the domain, which it didn't say anything about.

(311) a. Geen slimme man heeft elk boek gelezen.
no smart man has every book read
'No smart man read every book.'

b. Elke slimme man heeft niet elk boek gelezen. (wide scope for *elke*)
every smart man has not every book read
'Every smart man has not read every book.'

c. Er heeft een slimme man niet elk boek gelezen.
there has a smart man not every book read
'Some smart man has not read every book.'

Likewise, other entailment relations between quantifiers can be defined. An interesting subclass are quantifiers that allow for numerical reasoning, for example *at least n N* entails *at least n-m N* in increasing context, and *at least n+m N* in decreasing contexts (with n and m natural numbers). This reasoning is thus also sensitive to the monotonicity properties of the environment.

(312) a. Everyone read at least three books.

b. $\Rightarrow$ Everyone read at least two books.

c. $\not\Rightarrow$ Everyone read at least four books.

(313) a. No one read at least three books.
b. $\not\Rightarrow$ No one read at least two books.
c. $\Rightarrow$ No one read at least four books.

Since the information on the entailment direction is available locally, these effects are not difficult to capture. A rule just needs to allow substitution of Pred(X+incr+$\geq$n+Deps) by Pred(X+incr+$\geq$(n-1)+Deps)[10], and of Pred(X+decr+$\geq$n+Deps) by Pred(X+decr+$\geq$(n+1)+Deps).

The details of such entailments, as well as the exact choice of the logic to be implemented, still need to be worked out. The work of Seuren offers interesting possibilities, but is in itself also rather a research program, than a worked out system ready for implementation. Much about the logic of natural language is to date unknown. Therefore, it is not the ambition of this thesis to specify the complete inferential system. Rather, the intention is to develop a logical form to which insights on semantic inference can be applied.

4.7.4 Syllogisms

Sometimes it is necessary to combine pieces of information from different places that together entail the query. Keenan and Westerståhl (1997) gives the example in (314). One can easily see that it holds, by filling in for example *some* or a proper name for NP_{incr} and *no* for NP_{decr}.

(314) a. All socialists are vegetarians. NP_{incr} is a socialist.
$\Rightarrow \text{NP}_{incr}$ is a vegetarian.
b. All socialists are vegetarians. NP_{decr} is a vegetarian.
$\Rightarrow \text{NP}_{decr}$ is a socialist.

It can be considered a generalization of some of Aristotle's syllogisms, extending to other increasing and decreasing NPs. Several syllogisms fit into these two patterns, some after DeMorgan was applied to one or more of the propositions.

We can represent the pattern shown above more schematically as follows.

(315) a. Q_{incr} A is a B, every B is a C $\Rightarrow \text{Q}_{incr}$ A is a C
b. Q_{decr} A is a B, every C is a B $\Rightarrow \text{Q}_{decr}$ A is a C

The *every* part is extendable indefinitely, because *every x is a y* is transitive (if every x is a y and every y is a z then every x is a z).

(316) a. Q_{incr} A is a B, every B is a D, every D is a ... C $\Rightarrow \text{Q}_{incr}$ A is a C
b. Q_{decr} A is a B, every C is a D, every D is a ... B $\Rightarrow \text{Q}_{decr}$ A is a C

After the invention of first order logic, syllogisms were no longer needed as a tool for reasoning, since their validity follows from first order logic reasoning. But as we do not use first order logic in our representations, we need to derive the syllogisms in some

[10] The rule can be applied recursively.

other way. It is obvious that some rules are needed, but the application of these rules is complicated by the fact that it is unknown how many pieces of information will be needed to derive the inference searched for.

All syllogisms depend on a universal statement, either positive (*all*) or negative (*no*, which can be reanalyzed as *all not*), this is also the part that is extendable. Therefore, these patterns can be made accessible if chains of universals are compiled as part of the off-line representation of a text[11]. The text is searched to form these chains through the text. A chain can be headed by a different quantifier. This way a body of knowledge is compiled[12]. Chains go from predicates that share a variable to variables that share a predicate to predicates that share a variable etc. with conditions on the quantifiers, entailment properties and dependencies. Chains can be branching.

(317) A(X+incr+Q1+Dep1) – B(X+incr+Q1+Dep2) – B(Y+decr+every+Dep3) – C(Y+incr+every+Dep4) – C(Z+every…

Suppose we now want to know whether some linguists are vegetarians, and the text only tells us that some linguists are socialists and all socialists are vegetarians. After the entailment algorithm has not been able to find the information directly, it looks whether *some linguists* is connected to any universal chain that also contains vegetarians.

There will always be a finite number of chains of finite number of elements in a text. Chains may be circular, but also that is detectable and representable as such. Chaining is a commonly used technique in deriving instantiations of modus ponens in ontology languages, such as description logics (Jurafsky and Martin, 2000).

4.7.5 What else is needed for deriving entailments?

Suppose we have sufficiently accurate FLF representations on the sentential level, and an algorithm that derives entailments on the basis of these, do we then get all entailments that we want?

In the RTE challenges (see chapter one), it turned out that the main reason for missed entailments was a lack of background knowledge. We will assume here that we can use pre-compiled external sources, such as ontologies. Terms in the query can be looked up in order too enrich the query with extra information that can be found about these terms (see Bos and Markert (2006)).

Not all relevant information can be expected to be found this way. Consider the term *problem*. In Narrator it may happen that a user asks what problems others encountered when going back to work. Narratives that describe such problems may not always explicitly mention the word *problem* or a closely related term, especially when it will anyway be obvious to the human reader that what is being described is a problem. It is unlikely that it can be found in an ontology or even an encyclopedia what kind of thing counts as a problem for what or for who. A possible strategy is to try and harvest such

[11] Also generics are interesting in this respect, but they require techniques for non monotone reasoning.

[12] In order to combine information from different texts their universal chain information needs to be combined. (Some extra care is required here, since the contextually determined set that the universal applies to is more likely to be different.) In the context of the Narrator project, which is about the retrieval of single (parts of) documents, it is unlikely that this is needed.

information by collecting states of affairs that are explicitly described as problems from a corpus. Still, making useful generalizations on the basis of these data and then deciding what to search for in the narratives is far from trivial.

Reasoning about time and space will need background knowledge about some of the physical properties of our world. It needs to be investigated whether this can be covered by a set of FLF postulates.

Coreference resolution will also be important. The natural thing to do is to use an existing system or a combination of systems. Some systems are rather specialized, for example in recognizing when names in different formats refer to the same people.

Another discourse phenomenon not covered yet is that of discourse relations between sentences or clauses. Bosma (2008) gives an overview of the work in the field of annotation of discourse structure. Several taxonomies of coherence relation types exists using categories such as causality, justification, contrast, condition, and elaboration. There are annotated corpora automatic based on the Rhetorical Structure Theory (RST) of Mann and Thompson (1988). Marcu (1997) used such a corpus to develop an automatic RST annotation system. It mainly relies on layout and cue phrases.

In our system, explicitly indicated discourse relations within the sentence will in general be represented in the logical form. Relations between sentences will in some cases be a matter of coreference resolution. For example if a sentence begins with *therefore*, it needs to be established what *there* refers to. Discourse relations are however not always expressed explicitly. If I say for example *I'm going to bed. I'm tired.*, then anybody will immediately understand that I am going to bed *because* I am tired. Recognizing such implicit relations faces similar challenges as the *problem* problem above. It is therefore not clear to what extent the use of an existing system for annotating discourse relations will be helpful. The taxonomies probably will be of help to decide on useful and consistent representations for the cue phrases that are relevant to discourse relations.

4.8 Representation of text and hypothesis for entailment

This section is intended to summarize what has been discussed so far into a coherent picture as well as to give an outlook on what could not be covered here.

The texts are parsed off-line to get FLF representations for the sentences, representing their readings and equivalent forms. Every conjunct is annotated for the text, the sentence, the reading, and the alternative form it is part of. An alternative form of a reading can be obtained by applying DeMorgan or possibly some other defined operation that produces forms that are equivalent to the original one. All variables are distinct and have distinct names, unless they are bound by the same quantifier or stand in an anaphora relation. That is, fresh variable names need to be generated for every new sentence.

In order to combine information from different places it will be helpful to identify chains formed through predicates that share a variable and variables that share a predicate: P1(A)–P2(A)–P2(B)–P3(B)–P3(C). Especially when universal quantifiers are involved, such chains will help to find entailments.

In general it is important to do as much work as possible off-line, to make as many steps as possible in the direction of possible hypotheses. The remaining steps needed to check the entailment against such an elaborately represented text must be driven by the query/hypothesis. The hypothesis should determine what has to be searched for.

The hypothesis itself is also parsed and represented in FLF. The entailment algorithm derives search terms from the hypothesis; the hypothesis itself, its DeMorgan-equivalent normal form, minimal conjunctions that entail the hypothesis,. . . The latter can be derived by applying entailment rules, for example between different quantifiers. Changing a quantifier has consequences for more than one conjunct, therefore these rules cannot apply purely locally. Making use of the entailment properties in combination with, for example, an ontology can be applied to conjuncts in isolation. To look for evidence against the hypothesis, its negation can be derived (add negation to the dependency list of every conjunct and adjust the entailment properties), and the negations of its entailments.

The conjuncts of the hypothesis, in original or derived form, are then matched with conjuncts in the text. A conjunct P(Var1+Dir+Quant+[]) in the hypothesis, matches with a conjunct P(Var2+Dir+Quant+[]) in the text. If two conjuncts share the same variable in the hypothesis, their counterparts in the text must also share a variable in the same positions. After the preparatory steps, the actual matching is thus a matter of searching, plus a variable check.

If a dependent conjunct is found, while an independent one was searched for, it depends on the scopal operator in question whether the entailment holds or not. P(B+Dir+Quant+[A]) does not by default entail P(B+Dir+Quant+[]), but it can if A allows for it. This means that also conjuncts with more dependencies than in the hypothesis can be taken into consideration.

Below is an example that illustrates some aspects of inference on FLF. The hypothesis (h) in (319) is checked against a text (t) containing (318) as its 23^{rd} sentence. The matching conjuncts are indicated with the same symbol.

(318) *text*

a. Hoogstens twee mannen weten dat minstens drie vrouwen hard werken.
at most two men know that at least three women hard work
'At most two men know that at least three women work hard.'

b. man(A+decr+$>$2+[neg$_1$]):23:[1] &
proposition(B+nonm+the+[]):23:[1] &
evidence(B+incr+the+[], factive):23:[1] &
♠ woman(C+incr+$\geq$3+[B]):23:[1] &
♡ work(D+incr+some+[B,C]):23:[1] &
♢ event(D+incr+some+[B,C]):23:[1] &
♣ agent_of(D+incr+some+[B,C],C+incr+$\geq$3+[B]) &
attime(D+incr+some+[B,C],E):23:[1] &
manner_of(D+incr+some+[B,C], hard):23:[1] &
tense(D+incr+some+[B,C],pres):23:[1] &
know(F+decr+some+[neg$_1$]):23:[1] &
state(F+decr+some+[neg$_1$]):23:[1] &
experiencer_of(F+decr+some+[neg$_1$],A+decr+$>$2+[neg$_1$]):23:[1] &

theme_of(F+decr+some+[neg$_1$],B+incr+the+[]):23:[1] &
attime(F+incr+some+[neg$_1$],G):23:[1] &
tense(F+incr+some+[neg$_1$],pres):23:[1]

(319) *hypothesis*

a. Sommige vrouwen werken.
some women work
'Some women work.'

b. ♠ woman(X+incr+some_pl+[]) &
♡ work(Y+incr+some+[X]) &
♢ event(Y+incr+some+[X]) &
♣ agent_of(Y+incr+some+[X],X+incr+some_pl+[]) &
attime(X+incr+some+[X],Z) &
tense(Y+incr+some+[X],pres)

Every X in h corresponds to a C in t and every Y in h corresponds to a D in t. The quantifier on *woman* in h differs from the one in t. The entailment needs to be mediated by a rule: P(Var1+incr+some_pl+[]) is entailed by P(Var2+incr+>n+[]), for all n≥2. As for the dependencies, the conjuncts that are dependent on C in h, are dependent on X in t. That is fine. In addition, all the relevant conjuncts in t are dependent on B. This needs to be checked. Since B refers to a proposition marked as factive, it does not block entailment. The temporal information can in this case also be matched, but an exact match, especially of tense, is not always needed. (In the RTE task tense was ignored.)

The entailment is effective to the degree that the representation it operates on is analytical; we consider a representation to be fully analytical if every possible semantic consequence of a sentence is isolated in a closed proposition. The notion of entailment exploited is rather restrictive. It implies that nothing is entailed unless it is explicitly represented or can be deduced from representations with linear means. It is restricted to relations that are made explicit in the linguistic analysis. However, if additional knowledge is explicitly specified in a linguistic mode, the entailment mechanism may account for logical relations based on that knowledge. The restrictiveness of conjunctive entailment neglects other formal or informal conclusions a human being may derive from texts. Thus, applying conjunctive entailment provides, by necessity, a subset of documents supporting a certain query. Whether this still yields a useful and/or valuable level of retrieval, is to be evaluated in application.

The restrictiveness as such, however, is possibly balanced by the option of parameterizing conjunctive entailment. In particular, certain semi-veridical semantic operators like probability, which under strict conditions obscure propositions in their domain from entailment, can be handled more loosely as to allow inferences. That is, on a query p?, documents can be brought up that entail *mod*(p) for certain modal operators *mod*. The inference mechanism can be parameterized for this, or the results can be presented as supporting a weaker hypothesis. This applies to the different cases of embedding that are sometimes better ignored, such as modals used as hedges, or citations that the author can be assumed to agree with.

Similarly, if no sentence can be found that entails the hypothesis, but there is a sentence in the text that is entailed by the hypothesis, this may be returned as a weaker version of the hypothesis. For example, instead of *all men work*, you find *some men work*. Then one of the conjuncts in the hypothesis is not supported by the text, and therefore the hypothesis is not supported, but an entailment of the hypothesis is. Or, instead of *all men work* you find *all blond men work*, or instead of *all men work hard* you find *all men work*. In all these cases an output that reports on finding support for a weaker version of the hypothesis may be useful.

4.9 Examples of queries and inferences on real text

Throughout this chapter, we have seen examples of representations and inferences concerning the phenomena discussed. The examples were always constructed single sentences. Now we will briefly look at a real text that would be included in the Narrator system. It gives a clearer impression of the kind of stories Narrator is about and at the same time of the problems and opportunities of representing and inferencing on real text. On the next page you find the text that is used, found at http://www.kankerpatient.nl, followed by a rough translation.

The queries used for illustration have been constructed on the basis of the text. Natural language questions have been chosen as the query form. This is a natural type of input that is likely to profit from a deep semantic approach. Most, but not all are somewhat plausible as user input. The representations of those sentences from the text that are relevant to the queries are given, together with the inferences that are to be checked. In the FLF representations of the text sentences, variable names were used that consist of a letter and the number of the sentence, in order to generate enough different variable names for the whole text. In the hypotheses only letters are used and the variables are considered to be independent from those occurring elsewhere. In the process of matching hypothesis conjuncts against text conjuncts, variables unify. Sometimes extra white lines were introduced in the FLFs for reading convenience. Tense is largely ignored as the system for its representation is too poor to sensibly make use of it for inference with some form of temporal reasoning.

The examples do not necessarily represent results that other methods cannot get, but give an impression of the way our approach can be very precise.

Of course the new FLF format is only successful if the representations are in general of high quality, that is, if the right information is adequately encoded in it. Some limitations will immediately become apparent, as there are many aspects of semantics that lack the implementation of a systematic analysis.

The imaginary queries we will consider (some more likely to actually occur in the envisioned setting than others) are the following:

1. Did a breast cancer patient receive surgery and radiation?

2. Can someone who had axillary clearance still do cleaning?

3. What can carwax be used for?

Eind december 1992 werd bij mij borstkanker geconstateerd en onderging ik een borstsparende operatie en een okseltoilet. De behandelende chirurg meldde tussen neus en lippen door dat ik mijn rechterarm nooit meer functioneel zou kunnen gebruiken, dat ik misschien nog net mijn haar zou kunnen kammen. Verbijsterd hoorde ik de man aan, maar eigenwijs als ik was en nóg ben, heb ik dát verhaaltje naast me neergelegd.

Immers, ik moest thuis en op mijn werk toch gewoon kunnen functioneren, als kostwinner en alleenstaand moeder van een dochter van twaalf jaar. Na alle bestralingen, die voor mij alleen psychisch vervelend waren, heb ik mezelf drie maanden gegeven om bij te komen. Ik heb op mijn manier het verdriet en de boosheid verwerkt en ben vervolgens weer voor halve dagen aan het werk gegaan.

Ja, natuurlijk was ik moe, maar daaraan mocht ik niet toegeven. Met behulp van een fysiotherapeut ben ik mijn arm gaan trainen, want, zo dacht ik, die zit aan mijn lichaam en doet maar mee met de rest. Ook dat was heel vermoeiend, maar het werkte wel. Als ik van mijn werk thuiskwam, dook ik mijn bed in, in de hoop dat de vermoeidheid over zou gaan. Want daarna kwamen de boodschappen, de huishouding en de moeder-die-met-thee-klaar-zit als mijn dochter uit school kwam.

Dat heb ik een half jaar volgehouden en toen was het op! Ik heb me ziek gemeld en heb hulp gezocht bij een psychiater. Hij heeft me geleerd dat op je tenen lopen niet goed is, dat je zwak mag zijn, soms even niet kan of geen zin hebt om de dingen te doen die gebeuren moeten. Luister naar je lichaam, zei hij, en ik die dat altijd een 'geitenwollensokkenverhaal' vond, heb zijn raad aangenomen. Ik ontdekte trucjes die me in het huishouden helpen. Als ik daarover aan iemand vertel, verklaren ze me voor gek, maar wat kan mij dat schelen? Om een voorbeeld te noemen: als ik handwas heb, neem ik dat mee in bad en zit in het warme water op een relaxte manier de kledingstukken te wassen. Als de badkamer schoongemaakt moet worden, doe ik de muren terwijl ik zelf onder de douche sta en smeer vervolgens wat autowas op de tegels, zodat er de eerste weken geen kalkaanslag verschijnt. Kortom, ik ben het type 'luie huisvrouw' geworden. Als mensen zich storen aan de pluizen op het kleed is dat jammer, ze nemen dan zelf maar de stofzuiger ter hand. Ik vind het veel belangrijker dat ik lekker in mijn vel zit, me goed voel en daar heeft mijn dochter en de rest van de omgeving ook veel meer plezier van. En als ik heel erg moe ben, doe ik alleen de dingen die ik leuk vind, ga lekker een film kijken op bed, lees een boek, ga in de tuin zitten genieten...

Natuurlijk zijn er dagen dat het helemaal niet gaat, maar ik accepteer dat als iets dat erbij hoort. Alleen mijn werkgever was niet zo blij, ik kon mijn eigen werkzaamheden niet meer doen en toen ik ook nog een whiplash opliep tijdens de dienst - ik werkte bij de politie - en gedeeltelijk in de WAO terechtkwam, was dat een reden om mijn ontslag aan te vragen. In eerste instantie ben ik daar erg boos over geweest, voelde me afgedankt, maar eigenlijk ben ik nu dankbaar. Het geeft me meer tijd voor mezelf en daar word ik alleen maar beter van.

Ik hoop dat dit verhaal voor anderen bijdraagt aan de acceptatie van vermoeidheid. Denk er maar eens over na en laat dat 'moeten' eens varen. Je zult merken dat het helpt.

At the end of December 1992 I was diagnosed with breast cancer and I underwent a breast conserving surgery and an axillary clearance. The surgeon who treated me mentioned casually that I would never be able to functionally use my right arm again, that perhaps I could just still manage to brush my hair. Flabbergasted I listened to the man, but headstrong as I was and still am, I decided to ignore that little story.

After all, I would have to be able to function normally at home and at work, being a breadwinner and single mother of a twelve year old daughter. After all the radiations, which only bothered me psychologically, I gave myself three months to recover. In my own way I came to terms with the anger and sadness, and afterwards I started working again half-time.

Yes, of course I was tired, but I couldn't give in to that. With the help of a physiotherapist I started training my arm, because, so I thought, it is part of my body and it 'd better keep up with the rest. That was very tiring too, but it did work. When I came home from work I jumped into bed, hoping that the tiredness would go away. Because after that there was the grocery shopping, the household, and the mother-waiting-with-tea when my daughter came home from school.

I managed to continue like that for half a year and that was it! I called in sick and went to a psychiatrist for help. He taught me that walking on tiptoe is not good, that it's ok to be weak, that sometimes you just don't manage or don't feel like doing the things that need to be done. Listen to your body, he said, and I who always used to consider that something for sissies, accepted his advice. I discovered little tricks that help me in the household. When I tell someone about it, they think I'm crazy, but what do I care? To give an example: when I need to wash things by hand, I take them with me into the bath and I sit in the warm water washing those clothes in a relaxed way. when the bathroom needs to be cleaned, I clean the walls while I am taking a shower and then I use some car wax to wax the tiles, to prevent scale from appearing for the next few weeks. In short, I have become the 'lazy housewife' type. If people are bothered by fluff on the carpet they are free to get out the vacuum cleaner themselves. I find it much more important that I feel good and my daughter and the other people around me benefit much more from that too. And when I am really exhausted, I only do the things I like, I watch a nice movie in bed, read a book, sit down in the garden and enjoy. . .

Of course there are days that it doesn't go at all, but I accept that as something that's just part of it. Only my employer wasn't that happy, I couldn't do my own type of work anymore and when on top of that I got a whiplash while on duty - I worked for the police - and ended up relying partly on disability benefits, that was a reason to apply for my dismissal. At first I was very upset about that, felt discarded, but really I now am grateful. It gives me more time for myself and I only benefit from that.

I hope this story contributes for others to the acceptance of tiredness. Think about it, and let go of that ' have to'. You will see that it helps.

4. Can a someone who had breastcancer function normally at work?

The questions are transformed into hypotheses for which entailment is checked.

We start with the first query about whether a breast cancer patient had surgery and radiation. As it is a yes/no question the way to proceed is to look whether a hypothesis that would result in an affirmative answer is entailed by the text and whether a hypothesis that would result in a negative answer is entailed by the text. This kind of query could easily occur to select texts about particular types of treatment. This method of deriving a hypothesis (statement) from a question is common practice in question answering and is also used in the RTE challenge to transform question-answer pairs into text-hypothesis pairs for the data set. I assume algorithms are available.

So the hypothesis we will try to find evidence for or against and that will put this text in the yes or in the no category is the following (the result of parsing the Dutch equivalent of the statement derived from the question, which may contain verbs rather than nominalizations), assuming an implementation of coordination along the lines of Willis (2007):

```
person(A+incr+some+[])&
breastcancer(B+nonm+gen+[]) &
suffer_from(C+incr+some+[B]) &
state(C+incr+some+[B]) &
experiencer_of(C+incr+some+[B],A+incr+some+[]) &
theme_of(C+incr+some+[B],B+incr+gen+[]) &
event(D+incr+some+[]) &
operate(D+incr+some+[]) &
agent_of(D+incr+some+[],E) &
theme_of(D+incr+some+[],A) &
attime(D,F) &
tense(D,past) &

event(G+incr+some+[]) &
radiate(G+incr+some+[]) &
agent_of(G+incr+some+[],H) &
theme_of(G+incr+some+[],A) &
attime(G,I) &
tense(G,past)
```

Borstkankerpatient ‘breast cancer patient’ is here analyzed as ‘person who suffers from breast cancer’. In a system as Narrator it makes sense to include this in the the lexicon. Alternatively, this information can be looked up in an external source.

Evidence for the first part of the query is found in sentence number one. It needs to be specified somewhere that ‘i’ refers to a person[13].

[13] In cases where ‘i’ refers to personified things, one can either accept these things as persons in these contexts or alternatively ‘i’ can be specified to refer to a person only in absence of evidence to the contrary. The latter option needs a logic that is suitable for reasoning with this type of information.

"Eind december 1992 werd bij mij borstkanker geconstateerd en onderging ik een borstsparende operatie en een okseltoilet."
End of December 1992 I was diagnosed with breastcancer and I went through a breast conserving surgery and an axillary clearance.

year_1992(A1+nonm+the+[]) &
month_december(B1+nonm+the+[]) &
attime(B1+nonm+the+[],A1+nonm+the+[])&
end(C1+nonm+the+[]) &
related_to(C1+nonm+the+[],B1+nonm+the+[]) &
proposition(D1+incr+the+[]) &
breastcancer(E1+nonm+gen+[]) &
suffer_from(F1+incr+some+[D1,E1]) &
state(F1+incr+some+[D1,E1]) &
experiencer_of(F1+incr+some+[D1,E1],i) &
theme_of(F1+incr+some+[D1,E1],E1+incr+gen+[D1]) &
attime(F1+incr+some+[D1,E1],B1) &
tense(F1+incr+some+[D1,E1],past) &
ascertain(G1+incr+some+[]) &
event(G1+incr+some+[]) &
agent_of(G1+incr+some+[],H1) &
theme_of(G1+incr+some+[],D1+incr+the+[]) &
attime(G1+incr+some+[],C1+incr+the+[]) &
tense(G1+incr+some+[],past)&

breast(I1+decr+gen+[]) &
event(J1+incr+some+[]) &
operate(J1+incr+some+[]) &
agent_of(J1+incr+some+[],K1) &
theme_of(J1+incr+some+[],i) &
event(L1+incr+some+[]) &
conserve(L1+incr+some+[]) &
agent_of(L1+incr+some+[],J1+incr+some+[]) &
theme_of(L1+incr+some+[],I1+incr+gen+[]) &
attime(J1,E) &
tense(J1,past) &

armpit(M1+incr+some+[]) &
related_to(M1+incr+some+[],i) &
gland(N1+decr+every+[]) &
location_of(N1+decr+every+[],M1+incr+some+[]) &
event(O1+incr+some+[]) &
operate(O1+incr+some+[]) &

agent_of(O1+incr+some+[],P1) &
theme_of(O1+incr+some+[],i) &
subevent_of(O1+incr+some+[],Q1+incr+some+[]) &
event(Q1+incr+some+[]) &
remove(Q1+incr+some+[]) &
agent_of(Q1+incr+some+[],P1) &
theme_of(Q1+incr+some+[],N1+incr+every+[]) &
attime(O1,R1) &
tense(O1,past) &

Evidence for the second part is found in sentence number five. In one of its readings the theme of radiation is resolved as 'i', referring to the first-person narrator. If mechanisms to extract world knowledge are used, this might even be identified to be by far the most likely reading. The text says that the radiations were unpleasant for the 'i' referent. By extracting unambiguous cases from a corpus it may be concluded that it is most likely for radiation to be unpleasant for the person receiving it. For now we consider one legitimate reading to be enough to support the inference. Since the sentence is part of the same text as the previous one and no embedding in a direct quotation is involved, the 'i' referent of this sentence and the previous one is assumed to be the same. Therefore the entailment is recognized and the text is retrieved as providing an affirmative answer to the question asked in the query.

"Na alle bestralingen,[. . .] , heb ik mezelf drie maanden gegeven om bij te komen."
After all the radiations, I have given myself three months to recover.

event(A5+decr+every+[]):5:[1,2,3] &
radiate(A5+decr+every+[]):5:[1,2,3] &
agent_of(A5+decr+every+[],B5):5:[1,2] &
theme_of(A5+decr+every+[],C5):5:[1,3] &
agent_of(A5+decr+every+[],i):5:[3] &
theme_of(A5+decr+every+[],i):5:[2] &

month(D5+incr+three+[]):5:[1,2,3] &
give(E5+incr+some+[]):5:[1,2,3] &
event(E5+incr+some+[]):5:[1,2,3] &
source_of(E5+incr+some+[],i):5:[1,2,3] &
theme_of(E5+incr+some+[],D5+incr+three+[]):5:[1,2,3] &
goal_of(E5+incr+some+[],i):5:[1,2,3] &
attime(E5+incr+some+[],F5):5:[1,2,3] &
aspect(E5+incr+some+[],perf):5:[1,2,3] &
tense(E5+incr+some+[],pres):5:[1,2,3] &
property(G5+decr+the+[]):5:[1,2,3] &
recover(H5+incr+some+[G5]):5:[1,2,3] &
event(H5+incr+some+[G5]):5:[1,2,3] &

```
experiencer_of(H5+incr+some+[G5],i):5:[1,2,3] &
attime(H5+incr+some+[G5],I5):5:[1,2,3] &
purpose_of(E5+incr+some+[],H5+incr+the+[G5]):5:[1,2,3] &
after(A5+incr+every+[],E5+incr+some+[]):5:[1,2,3] &
```

The second query asked whether someone who had axillary clearance can still do cleaning. Also a reasonable question as the removal of the axillary glands may affect the functioning of the arm. Users may ask a question of this type when trying to find out whether (ex-)patients experienced any trouble with household activities and for example had to employ someone to do their cleaning. We will look for evidence for the following:

```
person(A+incr+some+[]) &
armpit(B+incr+some+[]) &
related_to(B+incr+some+[],A+incr+some+[]) &
gland(C+decr+every+[]) &
location_of(C+decr+every+[],B+incr+some+[]) &
event(D+incr+some+[]) &
operate(D+incr+some+[]) &
agent_of(D+incr+some+[],E) &
theme_of(D+incr+some+[],A+incr+some+[]) &
subevent_of(D+incr+some+[],F+incr+some+[]) &
event(F+incr+some+[]) &
remove(F+incr+some+[]) &
agent_of(F+incr+some+[],E) &
theme_of(F+incr+some+[],C+incr+every+[]) &
attime(D,G) &
tense(D,past) &

state(H+incr+some+[]) &
proposition(I+decr+the+[]) &
clean(J+incr+some+[I]) &
event(J+incr+some+[I]) &
agent_of(J+incr+some+[I],A+incr+some+[]) &
theme_of(J+incr+some+[I],K) &
attime(J+incr+some+[I],L) &
theme_of(H+incr+some+[],I+incr+the+[]) &
evidence(I+incr+the+[],possible) &
attime(H+incr+some+[],M) &
tense(H+incr+some+[],pres)
```

The evidence in this case is a bit less straightforward than in the previous example. The person undergoing axillary clearance is identified as the first-person narrator of the story as above. Now we have to establish whether this person is able to do cleaning. Sentence fourteen is about cleaning the bath room. The protagonist is said to ‘do’ the

walls. For the inference to be supported this 'do', which could in principle refer to any kind of activity, should be identified as cleaning. In this conditional relation to cleaning, this can be judged likely or at least compatible with the sentence, which is about cleaning and and the protagonist doing something and is therefore already likely to be relevant. It is reasonable to tune the system in such a way that this document is retrieved in the 'yes' list, but ranking below documents that contain more conclusive evidence, if there are any. The document can not be identified as strictly entailed, but, with a sufficiently advanced retrieval strategy, as relevant and compatible with a yes answer. A more powerful solution would be to consider *do* as anaphoric and resolving its reference. That would yield a real entailment.

"Als de badkamer schoongemaakt moet worden, doe ik de muren terwijl ik zelf onder de douche sta en smeer vervolgens wat autowas op de tegels, zodat er de eerste weken geen kalkaanslag verschijnt."
If the bathroom needs to be cleaned, I do the walls while I am taking a shower and then I spread some car wax on the tiles, so no lime scale appears in the first few weeks

wall(A14+decr+the+[]) &
do(B14+incr+some+[]) &
event(B14+incr+some+[]) &
agent_of(B14+incr+some+[],i) &
theme_of(B14+incr+some+[],A14+incr+the+[]) &
attime(B14+incr+some+[],C14) &
tense(B14+incr+some+[],pres) &
shower(D14+decr+the+[]) &
stand(E14+incr+some+[]) &
state(E14+incr+some+[]) &
theme_of(E14+incr+some+[],i) &
attime(E14+incr+some+[],F14) &
tense(E14+incr+some+[],pres) &
under(D14+incr+the+[],E14+incr+some+[]) &
while(B14+incr+some+[],G14) &
state(H14+incr+some+[]) &
proposition(I14+decr+the+[]) &
bathroom(J14+decr+the+[I14]) &
clean(K14+incr+some+[I14]) &
event(K14+incr+some+[I14]) &
agent_of(K14+incr+some+[I14],L14) &
theme_of(K14+incr+some+[I14],J14+incr+the+[I14]) &
attime(K14+incr+some+[I14],M14) &
theme_of(H14+incr+some+[],I14+incr+the+[]) &
evidence(I14+incr+the+[],necessary) &
attime(H14+incr+some+[],N14) &
tense(H14+incr+some+[],pres) &

```
if(H14+incr+some+[],B14+incr+some+[]) &

carwax(O14+incr+some+[]) &
tile(P14+decr+the+[]) &
smear(Q14+incr+some+[]) &
event(Q14+incr+some+[]) &
agent_of(Q14+incr+some+[],i) &
theme_of(Q14+incr+some+[],O14+incr+some+[]) &
goal_of(Q14+incr+some+[],P14+incr+the+[]) &
attime(Q14+incr+some+[],R14) &
tense(Q14+incr+some+[],pres) &

week(S14+decr+the+[]) &
first(S14+decr+the+[]) &
scale(T14+decr+some+[neg]) &
appear(U14+decr+some+[neg]) &
event(U14+decr+some+[neg]) &
theme_of(U14+decr+some+[neg],T14+decr+some+[neg]) &
attime(U14+decr+some+[neg],S14+decr+the+[]) &
tense(U14+decr+some+[neg],pres) &
atplace(P14+incr+the+[],U14+incr+some+[]) &

lead_to(V14+incr+some+[]) &
event(V14+incr+some+[]) &
cause_of(V14+incr+some+[],Q14+incr+some+[]) &
effect_of(V14+incr+some+[],U14+incr+the+[]) &
```

The third query is considerably less likely to occur naturally in the Narrator context, but is still interesting for purposes of illustration. It asks what car wax can be used for, a what-question for a change. I assume a representation like the following will be searched for.

```
state(A+incr+some+[]) &
proposition(B+decr+the+[]) &
carwax(C+incr+some+[B]) &
use(D+incr+some+[B]) &
event(D+incr+some+[B]) &
agent_of(D+incr+some+[B],E) &
theme_of(D+incr+some+[B],C+incr+some+[B]) &
lead_to(F+incr+some+[B]) &
event(F+incr+some+[B]) &
cause_of(F+incr+some+[B],D+incr+some+[B]) &
effect_of(F+incr+some+[B],G) &
attime(D+incr+some+[B],H) &
```

theme_of(A+incr+some+[],B+incr+the+[]) &
evidence(B+incr+the+[],possible) &
attime(A+incr+some+[],I) &
tense(A+incr+some+[],pres)

The answer is found in sentence number fourteen, already given above. Applying wax to tiles must be identified as an instance of use. I assume we can rely on ontologies here. The clause with *zodat* can be identified as introducing a purpose or effect. Rules for modality should make clear that when something is used, it can be used, but not the other way around. The use of this rule must be triggered by the occurrence of *can* in the hypothesis. On the basis the match that is obtained this way it is concluded that sentence fourteen contains the answer. Also notice that if the negation in the text scoped higher than it does, embedding the wax or the event, the inference could not be made and the document would not be retrieved. This illustrates the precision of detailed semantic analysis.

Our last example illustrates a case of embedding that blocks entailment. We have already seen before how we the first-person narrator in the story with the breast cancer patient in the query. So let us leave that step out here and just see what happens if we try to match 'Ik kon op mijn werk gewoon functioneren.' *I could function normally at work* against the text.

state(A+incr+some+[]) &
work(B+decr+the+[]) &
related_to(B+decr+the+[],i) &
proposition(C+decr+the+[]) &
property(D+decr+the+[C]) &
normal(E+incr+some+[C]) &
state(E+incr+some+[C]) &
theme_of(E+incr+some+[C],F+incr+some+[C]) &
function(F+incr+some+[C]) &
event(F+incr+some+[C]) &
agent_of(F+incr+some+[C],i) &
attime(F+incr+some+[C],G) &
theme_of(A+incr+some+[],C+incr+the+[]) &
evidence(C+incr+the+[],possible) &
attime(A+incr+some+[],H) &
tense(A+incr+some+[],past) &
atplace(B+incr+the+[],A+incr+some+[]) &

The relevant text sentence here is sentence number four. All conjuncts can be matched, except that the proposition introduced by *kunnen* 'can' in the text is dependent, whereas in the query/hypothesis it is not. The conjuncts proposition(C+decr+the+[]) and evidence(C+incr+the+[],possible) cannot immediately be matched with proposition(P4+decr+the+[O4]) and evidence(P4+incr+the+[O4],possible) respectively,

because the latter is dependent on O4. The nature of this dependency must be checked. The variable O4 belongs to the proposition introduced by *moeten* 'have to'. This embedding does not allow for the inference.

"[Immers,] ik moest thuis en op mijn werk [toch] gewoon kunnen functioneren, . . . "
After all, I would have to be able to function normally at home and at work, . . .

state(A4+incr+some+[]) &
state(B4+incr+some+[]) &
home(C4+decr+the+[]) &
proposition(D4+decr+the+[]) &
proposition(E4+decr+the+[D4]) &
property(F4+decr+the+[E4]) &
normal(G4+incr+some+[F4]) &
state(G4+incr+some+[F4]) &
theme_of(G4+incr+some+[F4],H4+incr+some+[E4]) &
function(H4+incr+some+[E4]) &
event(H4+incr+some+[E4]) &
agent_of(H4+incr+some+[E4],i) &
attime(H4+incr+some+[E4],I4) &
theme_of(A4+incr+some+[],E4+incr+the+[D4]) &
evidence(E4+incr+the+[D4],possible) &
attime(A4+incr+some+[],J4)] &
theme_of(B4+incr+some+[],D4+incr+the+[]) &
evidence(D4+incr+the+[],necessary) &
attime(B4+incr+some+[],K4) &
tense(B4+incr+some+[],past) &
atplace(C4+incr+the+[],B4+incr+some+[]) &

state(L4+incr+some+[]) &
state(M4+incr+some+[]) &
work(N4+decr+the+[]) &
related_to(N4+decr+the+[],i) &
proposition(O4+decr+the+[]) &
proposition(P4+decr+the+[O4]) &
property(Q4+decr+the+[P4]) &
normal(R4+incr+some+[Q4]) &
state(R4+incr+some+[Q4]) &
theme_of(R4+incr+some+[Q4],S4+incr+some+[P4]) &
function(S4+incr+some+[P4]) &
event(S4+incr+some+[P4]) &
agent_of(S4+incr+some+[P4],i) &
attime(S4+incr+some+[P4],T4) &
theme_of(L4+incr+some+[],P4+incr+the+[O4]) &

evidence(P4+incr+the+[O4],possible) &
attime(L4+incr+some+[],U4)] &
theme_of(M4+incr+some+[],O4+incr+the+[]) &
evidence(O4+incr+the+[],necessary) &
attime(M4+incr+some+[],V4) &
tense(M4+incr+some+[],past) &
atplace(N4+incr+the+[],M4+incr+some+[])&

The basis of reasoning with FLF is a matter of finding matching conjuncts. We have seen in the first example that these do not all need to come from the same sentence. A system of tracking referents is needed for many cases in which information from different sentences is combined. Additionally, successful inferencing relies on possibilities of expanding the query, through the use of ontologies and, for example, rules of modality. Many aspects of the representation that are independent of the FLF format remain important. For example in the third query, the representation of purpose introduced in one case by *use for* and in the other case by *zodat* was in both cases the same. If it were not, additional rules would be necessary. Because complex relations are concerned, this is not straightforwardly covered by an ontology. Of course, some information that is needed for inference cannot be included in the representations which are computed compositionally, because it is dependent on interpretation in context, with the help of world knowledge. There is however no reason why strategies that make guesses about such things possible, for example by using harvested knowledge, should not be able to be integrated with the present approach.

All these additional mechanisms are necessary in other logical approaches too, except maybe referent tracking, which might be already covered in DRT. The advantage of FLF is that the basic logical operation is simply the matching of conjuncts.

4.10 Conclusions and future research

This chapter introduced Flat Logical Form (FLF), a semantic output format of our semantic parser (and generator) for Dutch, which is specifically designed to facilitate inferencing. Quantifiers and their properties are coded on the variables they bind. What remains is a conjunction of predicates over variables with all quantificational properties and dependencies marked on those variables, subject to conjunctive entailment. In order to make use of the quantificational properties for entailment, rules that operate on these properties are to be formulated. After a number of preparatory steps on the text and the hypothesis, checking entailment is a matter of searching in a list of conjuncts.

I have introduced a rather coarse-grained system for computing effective entailment directions. When continuous quantifiers are analyzed in terms of increasing and decreasing ones, no additional property is needed to obtain the continous entailment pattern. Consistent and complete entailments can also be obtained by other means. Further research is needed to determine how to best make (anti-)additive and (anti-)multiplicative entailment patterns available.

A promising aspect of FLF that is yet to be explored is that the representation

can be viewed as a network. The conjuncts are separate, but interlinked units, of which the order does not matter. Therefore, FLF representations are potentially more representative for the way meanings are represented in the human brain, than first-order logic representations are.

FLF is expected to facilitate entailment-based retrieval in the Narrator system. FLF offers semantic representation at a level of specification that renders obsolete deep inspection of the formula for the sake of automated inference: the computation of the majority of logical dependencies has been transferred to the off-line semantic representation. The surface representation only makes use of conjunction and to a limited extent disjunction. Entailment directions and dependencies induced by quantifiers and different kinds of operators, including negation, are all coded on the variables in an explicit and localized way. Entailments are mediated by simple rules that operate on the encoded properties. Of course, as is the case in any kind of deep semantic analysis, countless phenomena are still waiting for improved analyses and representations. The FLF approach is expected to shed an interesting new light on at least some of the known challenges. More research is needed to further develop FLF and experiment with applications. Yet, I hope to have shown that FLF is a promising new way of semantic representation.

Conclusions and outlook

This thesis investigated meaning representation, with as a main criterion that it should be optimized for computing entailments. A way of representation has been developed, which contains detailed information, accounts for a number of entailments straightforwardly, but is simple in structure. The resulting format can capture more natural language meanings than first-order logic and at the same time is easier to handle computationally, because of its flat structure, which properties relevant for entailment can be read off locally.

The Narrator project offered a concrete aim to work towards: the retrieval of narrative documents about personal experiences related to breast cancer, to be made available to fellow-sufferers. The differences that determine how relevant a story is will often be a matter of details. (For example, a user may be looking for particular daily life situations or particular kind of emotional reaction.)

The basis that the research presented here started out from was the existing Delilah system (to be used to parse narratives), which parses (and generates) Dutch sentences. It is built on a variant of multi-modal combinatory categorial grammar designed to deal with the kind of discontinuities that occur in Dutch. Lexical items are feature-value matrices (in graph format). One of the features has as a value the category that is targeted by the grammar rules. Lexical items can compose to form a bigger unit, if a grammar rule allows for it and also their other features can be unified with each other. The working lexicon is fast but static, as it is completely compiled and indexed before the parsing session. This means that recursive lexical rules are not possible. (Since the compiling is relatively time-consuming, making adjustments and recompiling during a session would completely undo the speed benefit.) The semantic analysis works with a nested storage mechanism. An important feature of Delilah is its ability to deal with extended lexical units. It allows for the local semantics of fixed arguments to be ignored and for the whole phrase to get a dedicated meaning, lexically determined at its highest level.

The current system has two main limitations in terms of structural coverage. First, long extraction of VP adverbials is problematic. I have shown how this can be remedied by changing the category of the adjunct. A problem of the same nature occurs with extraposed relative clauses. Second, multiple coordination and ellipsis are at present not covered. This looks like a solvable problem, too, but not within the scope of the present work. Also, a number of robustness issues need to be solved. The system as it is, is suitable for a proof of concept of new ideas about semantic representation, but not for larger scale testing. Implementations show that things work in principle, e.g. that the desired representations are obtained for particular types of constructions.

I discussed the following minor improvements to the system, as part of chapter two: 1) I proposed a way of processing the stores that prevents generating the same reading

twice. 2) I argued for switching to Dutch concept names instead of the present English ones. 3) The separable negation of *geen* is now stored to account for split scope effects. (Modal verbs and scopal adverbs do not need to be stored.) 4) I showed how VP adjuncts can be selected if their category is no longer *x/x*, but a special category *a*, which is selected as an optional argument.

The larger scale improvements made, were the implementation of event semantics (discussed in chapter three) and the development of Flat Logical Form (the topic of chapter four). Both contribute to making the representations flatter. Event semantics also makes them 'deeper', i.e. more informative. Flat Logical Form to some extent does so as well, as it makes non-standard quantifiers less problematic to represent.

A form of neo-Davidsonian event semantics was implemented. In this implementation, the adding of a logical predicate *event* preserves the information included in the representations in the literature. The predicate *event* provides ontological information, being a hypernym of the verb. The event is introduced with an existential quantifier, and in verbs, it is introduced in the store, so it can in principle raise for wider scope. It was shown that in some cases subevents are needed to account for certain entailments. The issue of participant roles, turned out to be a complicated one. There is no ready theory of participant roles that is suitable for our purposes, though in both Dowty's (1991) theory and the FrameNet approach (Baker et al., 1998) entailments played a role, and both have interesting aspects. Starting out with considering participant roles verb specific and then making generalizations seems to be a useful approach.

In implementing event semantics for verbs, I distinguished the following classes: verbs that introduce one event and a number of participants that get roles, auxiliaries and epistemic modals, which do not introduce an event of their own, verbs that do introduce an event but also take a complement with internal event structure, verbs in which the events have results, and verbs that require an analysis with parallel subevents, because of the involvement of their participants.

I also implemented event semantics for nominalizations. It accounts for entailment relations between nominalizations and verbs. The participants are either expressed or silent. They can be expressed as arguments or be interpreted through binding. In any case, they must always be already introduced in the semantics of the nominal, because they can be bound in discourse and because they get participant roles. For support verb constructions it was shown that they need more than just binding of a participant.

States were discussed separately, because they are more controversial. I reached the conclusion that Davidsonian states are the right choice for attributive adjectives and their derived nouns, and also for stative verbs. Stative light verb constructions, previously treated as units under the ELU approach, are reanalyzed, using states and binding of the participant. This accounts better for their flexibility.

Flat Logical Form (FLF) is the new approach to meaning representation developed in this thesis. In FLF, conjunctions are the highest operators. Each conjunct is a predicate over a variable (or two variables in case of a two-place predicate). Each variable is annotated with the effective entailment direction that applies to its predicate, the quantifier that binds it, and the quantifiers and operators (including negation) it is dependent on. The effective entailment direction can be computed, because dependence

on decreasing quantifiers and negation, which also brings in a decreasing property, is marked on the variable. For this purpose the basic theorems by Zwarts (1986) were used, rather than Kas's 1993 complex calculus. Also the behavior of non-monotone quantifiers was incorporated. Intensional environments and scope are also handled in terms of dependency. The quantifiers can license entailment rules that are also sensitive to the entailment direction of the environment. A two-place predicate is in the scope of the quantifiers of both its variables. Therefore, the effective entailment direction that comes with the lowest of the two is the one that applies to the direct environment of the predicate. I argued that most decreasing quantifiers can be analyzed, like *geen*, with a separable negation. This accounts for them giving rise to split scope effects. Marking every conjunct for the sentence and reading it is part of makes it possible to give a compact representation of all readings of a sentence. I showed that the DeMorgan rules can be reformulated to work on FLF. DeMorgan for quantifiers is used to compute an equivalent form in which the negation cannot be lowered further. A rule of negation elimination cancels two adjacent negations against each other. Consistent and complete entailment patterns can be covered in a DeMorgan-like way, in spite of the fact that the fine-grained calculus is not used. Checking entailment basically becomes a matter of finding matching conjuncts. In non-increasing context, however, there are additional requirements on certain conjuncts occurring together. The main trick of FLF is that two pieces of information that are normally implicitly encoded in the structure of logical formulas, namely the effective entailment direction resulting from monotonicity and dependency on higher operators, are made explicitly and locally available in FLF. Search strategies will need to be developed to reproduce syllogism-like reasoning. I already proposed the preparation of universal chains. Patterns like *Name is X, Name does Y* $\models$ *some X does Y* do not need such preparation, because the number of steps will be limited. Still, a good strategy is needed to perform the different steps. This will be part of the search algorithm, yet to be developed, that performs the actual textual entailment checking by searching the FLF representations of the text to find support for the hypothesis.

I concluded that our notion of entailment or inference should include both strict entailment, conventional implicature and presuppositions. Computable cases of conversational implicature can be useful, but do not always make reliable inferences. Paraphrases that are not instances of these are not to be considered entailments. This does not necessarily mean that they cannot be recognized and used (in some cases they may satisfy a weaker hypothesis), but they are not to be confused with entailments.

Working on a system like Delilah and its semantic representations basically means working on the semantic analysis of a whole language. It is therefore not difficult to come up with ideas for further research. I will mention a few which naturally connect to what I have done. To really see inference on FLF at work, a strategy to check entailments needs to be developed. A logic of quantifiers is to be built into the entailment rules that are used. FLF offers the possibility to derive entailments from non-standard quantifiers. Ideally, a comprehensive 'natural' logic of quantifiers is needed. Next to that, several aspects of FLF itself need to be worked out further, such as the representation of different operators and of proper names, the composition of complex quantifiers, and coordination. Since

the FLF notation is so different from the conventional one, it invites for re-examining natural language semantic phenomena to see how exactly they are best represented in FLF, and if that leads to new insights. As it looks now, I expect that most of these topics will lead to satisfactory outcomes, which confirm the usefulness of FLF.

Independently of FLF, there are many other aspects of the semantic analysis that could be improved. For instance, many aspects of event analysis could use improvement. Much theoretical work is still being done in this field. Likewise, there is still little consensus about the semantics of degrees and degree modifiers. I touched upon the issue in the context of stative light verb constructions. It is relevant for all kind of constructions involving adjectives. Tense and temporal expressions need a coherent approach in which they define the timing of events in discourse. Here one could build on the results of the TimeML project (Pustejovsky et al., 2003). All of these are complex issues with many unsolved problems at the theoretical level.

On as somewhat more basic level, there are some issues that should make the coverage of the system more complete. Resolving these will enable larger scale testing, which will be good for trying out the benefits of the new advances in practice. Ellipsis and multiple coordination need to be implemented. Some preparatory work for that has already been done. There are a few other grammatical gaps to be filled as well, such as syntactic nominalizations. The lexicon also needs to be extended. There are plans to import the lexical entries of the Alpino parser, which covers a fairly large corpus of newspaper text (Bouma et al., 2001). An algorithm developed for guessing categories for unknown words needs to be implemented.

The Recognizing Textual Entailment challenge shows that progress is being made in the field of entailment engines (Giampiccolo et al., 2007). In the RTE challenges it becomes clear that one of the biggest remaining problems is the lack of word and world knowledge. (The boundary between the two is not always clear.) Also the recurring theme in the second chapter about how deep the analysis should go, is essentially about how much of this kind of knowledge should be incorporated in the representation. Ultimately, I think knowledge representations will always be poorer when the knowledge is not grounded in real experience consisting of interaction with others and with the world, than when it is. This holds even if the knowledge is harvested from large amount of data and and regardless of whether it is recorded explicitly or implicitly. So far systems used for textual entailment and similar applications only have access to linguistic data, sometimes manipulated, annotated or translated to some form of logic. None are able to perceive the non-linguistic world which the texts are about. It is not surprising if such a system can simulate understanding only to a limited extent.

Real-world experience however does not solve the problem of meaning representation. Steels and Bleys (2005) did pioneering experiments with embodied agents that developed their own language by playing language games with each other. The language of these agents is therefore grounded in real-world experience. At the same time, the agents do make use of logic-based meaning representations. Their fluid construction grammar contains predicates and variables. The predicates refer to complex non-static concepts, formed through observing things in the world and communicating about them. In this state-of-the-art fundamental research into how linguistic expressions and concepts

are formed hand in hand, taking all kind of factors about embodied communication in a physical world into account, semantic representations are still an important issue. This suggest that the main challenges of semantic representations are not an artifact of the limitations of purely symbolic systems, like the one described here. How to represent linguistic meanings remains an important line of research.

Bibliography

Alexiadou, A. (1997). *Adverb Placement: A Case Study in Antisymmetric Syntax*. Amsterdam: John Benjamins Publishing Company.

Alshawi, H. (1992). *The Core Language Engine*. Cambridge, Massachusetts: MIT Press.

Altmann, G. (1999). Thematic role assignment in context. *Journal of Memory and Language 41*(1), 124–145.

Amsili, P. and N. Hathout (1996). Computational semantics of time/negation interaction. In *Proceedings of the 16th conference on Computational linguistics*, Copenhagen, Denmark, pp. 29–34. Association for Computational Linguistics.

Anderson, C. (2004). *The Structure and Real-Time Comprehension of Quantifier Scope Ambiguity*. Ph. D. thesis, Northwestern University.

Arsenijević, B. (2006). *Inner Aspect and Telicity*. Ph. D. thesis, Leiden University.

Asher, N. (1993). *Reference to Abstract Objects in Discourse*. Dordrecht: Kluwer Academic Publishers.

Austin, J., S. Engelberg, and G. Rauh (2004). Current issues in the syntax and semantics of adverbials. In J. Austin, S. Engelberg, and G. Rauh (Eds.), *Adverbials: The Interplay Between Meaning, Context, and Syntactic Structure*, pp. 1–44. John Benjamins Publishing Company.

Badia, T. and R. Saurí (1998). The representation of syntactically unexpressed complements to nouns. In F. Busa, I. Mani, and P. Saint-Dizier (Eds.), *(Workshop on) The Computational Treatment of Nominals*, COLING-ACL, Montreal, Quebec, pp. 1–9.

Baker, C. F., C. J. Fillmore, and J. B. Lowe (1998). The Berkeley FrameNet project. In *Proceedings of COLING ACL*, Montreal, Quebec, pp. 86–90.

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*, Venice, Italy, pp. 1–9.

Bartsch, R. (1986). On aspectual properties of Dutch and German nominalizations. In V. Lo Cascio and C. Vet (Eds.), *Temporal Structure in Sentence and Discourse*, Number 5 in Groningen-Amsterdam Studies in Semantics, pp. 7–39. Dordrecht: Foris Publications.

Barwise, J. and R. Cooper (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy 4*(2), 159–219.

Beghelli, F. and T. Stowell (1997). Distributivity and negation: The syntax of each and every. In A. Szabolcsi (Ed.), *Ways of scope taking*, pp. 71–107. Dordrecht: Kluwer Academic Publishers.

Berners-Lee, T., J. Hendler, and O. Lassila (2001). The Semantic Web. *Scientific American 284*(5), 28–37.

Blackburn, P. and J. Bos (2003). Computational Semantics. *Theoria: Revista Trimestral de Teoria, Historia y Fundamento de la Ciencia 18*(46), 27–45.

Blackburn, P. and J. Bos (2005). *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Stanford, California: CSLI Publications.

Blackburn, P. and J. Bos (forthcoming). *Working with Discourse Representation Theory: An Advanced Course in Computational Semantics*. Stanford, California: CSLI Publications.

Blackburn, P., J. Bos, M. Kohlhase, and H. de Nivelle (2001). Inference and computational semantics. In H. Bunt, R. Muskens, and E.Thijsse (Eds.), *Computing Meaning*, Volume 2, pp. 11–28. Dordrecht: Kluwer Academic Publishers.

Bobrow, D., C. Condoravdi, R. Crouch, V. de Paiva, L. Karttunen, T. King, R. Nairn, L. Price, and A. Zaenen (2007). Precision-focused textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Rochester, New York, pp. 16–21.

Bos, J. (1996). Predicate logic unplugged. In *In Proceedings of the 10th Amsterdam Colloquium*, pp. 133–143.

Bos, J. (2005). Towards wide-coverage semantic interpretation. In *Proceedings of the Sixth International Workshop on Computational Semantics IWCS-6*, Tilburg, pp. 42–53.

Bos, J. (2008). Introduction to the shared task on comparing semantic representations. In J. Bos and R. Delmonte (Eds.), *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Volume 1 of *Research in Computational Semantics*, pp. 257–261. College Publications.

Bos, J., S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier (2004). Wide-coverage semantic representations from a CCG parser. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, Geneva, Switzerland, pp. 1240. Association for Computational Linguistics.

Bos, J., J. R. Curran, and E. Guzzetti (2007). The pronto QA system at TREC 2007: Harvesting hyponyms, using nominalisation patterns, and computing answer cardinality. In *Proceedings of TREC 2007*.

Bos, J. and K. Markert (2006). Recognising textual entailment with robust logical inference. *Lecture Notes in Computer Science 3944*, 404.

Bos, J., Y. Mori, B. Gambäck, M. Pinkal, C. Lieske, and K. Worm (1996). Compositional semantics in Verbmobil. In *Proceedings of the 16th conference on Computational Linguistics*, Volume 1, Copenhagen, Denmark, pp. 131–136. Association for Computational Linguistics.

Bosma, W. (2008). *Discourse Oriented Summarization*. Ph. D. thesis, University of Twente.

Bouma, G. and G. van Noord (1994). Constraint-based categorial grammar. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, Las Cruces, New Mexico, pp. 147–154. Association for Computational Linguistics.

Bouma, G., G. van Noord, and R. Malouf (2001). Alpino: Wide-coverage computational analysis of Dutch. In *Computational Linguistics in The Netherlands 2000*, pp. 45–59.

Broekhuis, H. (1999). Adjectives and adjective phrases. Working Paper 2, University of Tilburg.

Bunt, H. (1985). *Mass Terms and Model-Theoretic Semantics*. New York: Cambridge University Press.

Bunt, H. (2007). Underspecification in semantic representations: Which technique for what purpose? In H. Bunt and R. Muskens (Eds.), *Computing Meaning*, Volume 3 of *Studies in Linguistics and Philosophy*, pp. 55–86. Springer.

Burnage, G. (1990). *CELEX — A Guide for Users*. Centre for Lexical Information, Nijmegen.

Carlson, G. (2001). Thematic roles and the individuation of events. In S. Rothstein (Ed.), *Events and Grammar*, pp. 35–51. Dordrecht: Kluwer Academic Publishers.

Chambers, N., D. Cer, T. Grenager, D. Hall, C. Kiddon, B. MacCartney, M. de Marneffe, D. Ramage, E. Yeh, and C. Manning (2007). Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Rochester, New York, pp. 165–170.

Chierchia, G. and S. McConnell-Ginet (2000). *Meaning and Grammar: An Introduction to Semantics*. Cambridge, Massachusetts: MIT Press.

Chomsky, N. (1981). *Lectures on Government and Binding*. Dordrecht: Foris Publications.

Chomsky, N. (1993). A minimalist program for linguistic theory. In K. Hale and S. Keyser (Eds.), *The View From Building 20*, pp. 1–52. Cambridge, Massachusetts: MIT Press.

Cinque, G. (1995). Adverbs and the universal hierarchy of functional projections. *GLOW Newsletter 34*, 14–15.

Cinque, G. (2004). Issues in adverbial syntax. *Lingua 114*(6), 683–710.

Cooper, R. (1975). *Montague's Semantic Theory and Transformational Syntax*. Ph. D. thesis, University of Massachusetts.

Copestake, A. and D. Flickinger (2000). An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pp. 591–598.

Copestake, A., D. Flickinger, C. Pollard, and I. Sag (2005). Minimal Recursion Semantics: An introduction. *Research on Language & Computation 3*(4), 281–332.

Coppen, P. (1995). A new version of the AMAZON/CASUS system. In P. de Haan and N. Oostdijk (Eds.), *Proceedings of the Department of Language and Speech*, Volume 18, Nijmegen, pp. 85–90. University of Nijmegen.

Cremers, C. (1983). On two types of infinitival complementation. In F. Heny and B. Richards (Eds.), *Linguistic Categories: Auxiliaries and Related Puzzles*, pp. 169–221. Dordrecht: Reidel.

Cremers, C. (1993). *On Parsing Coordination Categorially*. Ph. D. thesis, Leiden University.

Cremers, C. (1999a). A note on categorial grammar, disharmony and permutation. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway, pp. 273–274. Association for Computational Linguistics.

Cremers, C. (1999b). Formalizing the syntax. Unpublished manuscript, available at http://www.delilah.eu/.

Cremers, C. (2001). Why pluralities don't mean a thing. In *Quitte ou double sens. Articles sur l'ambiguïté offerts à Ronald Landheer*, pp. 33–43. Amsterdam: Atlanta/Rodopi.

Cremers, C. (2002). ('n) Betekenis berekend. *Nederlandse Taalkunde 7*, 375–395.

Cremers, C. (2004). Modal merge and minimal move for dislocation and verb clustering. *Research on Language & Computation 2*(1), 87–103.

Croft, W. (2001). *Radical Construction Grammar*. New York: Oxford University Press.

Crouch, R., L. Karttunen, and A. Zaenen (2006). Circumscribing is not excluding: A response to Manning. Available at http://www2.parc.com/istl/members/karttune/publications/reply-to-manning.pdf.

Crouch, R., R. Sauri, and A. Fowler (2005). AQUAINT pilot knowledge-based evaluation: Annotation guidelines. Available at http://www2.parc.com/istl/groups/nltt/papers/aquaint_kb_pilot_evaluation_guide.pdf.

Dagan, I., O. Glickman, and B. Magnini (2006). The PASCAL Recognising Textual Entailment challenge. In J. Quiñonero Candela, I. Dagan, B. Magnini, and F. d'Alché Buc (Eds.), *Machine Learning Challenges*, Volume 3944 of *Lecture Notes in Computer Science*, pp. 177–190. Springer.

Dahl, D. A., M. S. Palmer, and R. J. Passonneau (1987). Nominalizations in PUNDIT. In *Proceedings of the 25th annual meeting on Association for Computational Linguistics*, Stanford, California, pp. 131–139. Association for Computational Linguistics.

Dalrymple, M. (2001). *Lexical Functional Grammar*. San Diego, California: London Academic Press.

Dang, H., D. Kelly, and J. Lin (2007). Overview of the TREC 2007 Question Answering Track. In E. Voorhees and L. Buckland (Eds.), *The Sixteenth Text REtrieval Conference Proceedings 2007*, Number 500-274 in Special Publication, Gaithersburg, Maryland. NIST.

Davidson, D. (1967). The logical form of action sentences. In N. Rescher (Ed.), *The Logic of Decision and Action*, pp. 81–95. Pittsburgh, Pennsylvania: University of Pittsburgh Press.

de Swart, H. (1996). Scope ambiguities with negative quantifiers. In C. von Heusinger and U. Egli (Eds.), *Proceedings of the Konstanz Workshop: Reference and Anaphorical Relations*, Number 79 in Arbeitspapier, pp. 145–164. Fachgruppe Sprachwissenschaft der Universität Konstanz.

Deerwester, S., S. Dumais, G. Furnas, T. Landauer, and R. Harshman (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science 41*(6), 391–407.

Dölling, J. (2005). Copula sentences and entailment relations. *Theoretical Linguistics 31*(3), 317–329.

Dowty, D. (1991). Thematic proto-roles and argument selection. *Language 67*, 547–619.

Ebert, C. (2005). *Formal Investigations of Underspecified Representations*. Ph. D. thesis, King's College.

Egg, M. (2004). Mismatches at the syntax-semantics interface. In S. Müller (Ed.), *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar*, Stanford, California, pp. 119–139. CSLI Publications.

Egg, M. (2005). Against opacity. In F. Richter and M. Sailer (Eds.), *Proceedings of he ESSLLI'05 Workshop on Empirical Challenges and Analytical Alternatives to Strict Compositionality*, Heriot-Watt University Edinburgh, Scotland, pp. 120–132.

Engelberg, S. (2004). Lexical event structures for verb semantics. *Journal of Language and Linguistics 3*(1), 62–108.

Engelberg, S. (2005). Kimian states and the grammar of predicative adjectives. *Theoretical Linguistics 31*, 331–347.

Erbach, G. and B. Krenn (1993). Idioms and support-verb constructions in HPSG. CLAUS Report 28, Universität des Saarlandes.

Fillmore, C., C. Baker, and H. Sato (2004). FrameNet as a "Net". In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Volume 4, Lisbon, Portugal, pp. 1091–1094. ELRA.

Fillmore, C. J., C. F. Baker, and H. Sato (2002). Seeing arguments through transparent structures. In *Proceedings of LREC 2002*, Las Palmas de Gran Canaria, Spain, pp. 787–791.

Fodor, J. (1982). The mental representation of quantifiers. In S. Peters and E. Saarinen (Eds.), *Processes, Beliefs, and Questions: Essays on Formal Semantics of Natural Language and Natural Language Processing*, pp. 129–164. Dordrecht: Reidel.

Fyodorov, Y., Y. Winter, and N. Francez (2000). A natural logic inference system. In *Inference in Computational Semantics ICoS-2 Proceedings*.

Gehrke, B. (2008). *Ps in Motion: On the Semantics and Syntax of P Elements and Motion Events*. Ph. D. thesis, Utrecht University.

Geurts, B. and F. van der Slik (2005a). Monotonicity and Processing Load. *Journal of Semantics 22*(1), 97–117.

Geurts, B. and F. van der Slik (2005b). Ups and downs in syllogistic reasoning. Unpublished manuscript, available at http://www.linguistics.pomona.edu/LGCS121Spring2005/Reading/Geurtsupsdowns.pdf.

Giampiccolo, D., B. Magnini, I. Dagan, and B. Dolan (2007). The third PASCAL Recognizing Textual Entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Rochester, New York, pp. 1–9. Association for Computational Linguistics.

Grefenstette, G. and S. Teufel (1995). Corpus-based method for automatic identification of support verbs for nominalizations. In *Proceedings of the 7th Conference of the European Chapter of the ACL*, Dublin, Ireland, pp. 27–31.

Grimshaw, J. (1990). *Argument Structure*. Cambridge, Massachusetts: MIT Press.

Groenendijk, J. and M. Stokhof (1991). Dynamic Predicate Logic. *Linguistics and Philosophy 14*(1), 39–100.

Herburger, E. (2000). *What Counts: Focus and Quantification*. Cambridge, Massachusetts: MIT Press.

Hickl, A. and J. Bensley (2007). A discourse commitment-based framework for recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Rochester, New York, pp. 171–176. Association for Computational Linguistics.

Higginbotham, J. (1985). On semantics. *Linguistic Inquiry 16*(4), 547–593.

Higginbotham, J. (2000). On events in linguistic semantics. In J. Higginbotham, F. Pianesi, and A. Varzi (Eds.), *Speaking of Events*, pp. 49–79. Oxford, New York: Oxford University Press.

Hijzelendoorn, M. and C. Cremers (2007, December). An Object-Oriented and Fast Lexicon for Semantic Generation. Paper presented at the 18th meeting of Computational Linguistics in the Netherlands (CLIN 2007), Nijmegen.

Hinrichs, E. (1986). Temporal anaphora in discourses of English. *Linguistics and Philosophy 9*(1), 63–82.

Hoekstra, T. (1999). Parallels between nominal and verbal projections. In D. Adger (Ed.), *Specifiers: Minimalist Approaches*, pp. 163–187. Oxford University Press.

Hoenkamp, E. (2003). Unitary operators on the document space. *Journal of the American Society for Information Science and Technology 54*(4), 321–334.

Hull, R. and F. Gomez (1996). Semantic interpretation of nominalizations. In *Proceedings of AAAI-96*, Portland, Oregon, pp. 1062–1068.

Jackendoff, R. (1972). *Semantic Interpretation in Generative Grammar*. Cambridge, Massachusetts: MIT Press.

Jackendoff, R. (1983). *Semantics and Cognition*. Cambridge, Massachusetts: MIT Press.

Jackendoff, R. (1987). The status of thematic relations in linguistic theory. *Linguistic inquiry 18*(3), 369–411.

Jacobs, J. (1980). Lexical decomposition in Montague-Grammar. *Theoretical Linguistics Berlin 7*(1-2), 121–136.

Jaspers, D. (2005). *Operators in the Lexicon: On the Negative Logic of Natural Language*. Ph. D. thesis, Catholic University of Brussels.

Johnson, C. and C. J. Fillmore (2000). The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, San Francisco, California, pp. 56–62. Morgan Kaufmann Publishers Inc.

Johnson-Laird, P. (1969). On understanding logically complex sentences. *The Quarterly Journal of Experimental Psychology 21*(1), 1–13.

Jurafsky, D. and J. Martin (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Upper Saddle River, New Jersey: Pearson/Prentice-Hall.

Kamp, H. and U. Reyle (1993). *From Discourse to Logic: Introduction to Model-Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Studies in Linguistics and Philosophy. Dordrecht: Kluwer Academic Publishers.

Karttunen, L. and A. Zaenen (2005). Veridicity. In G. Katz, J. Pustejovsky, and F. Schilder (Eds.), *Annotating, Extracting and Reasoning about Time and Events*, Number 05151 in Dagstuhl Seminar Proceedings. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI). Available at http://drops.dagstuhl.de/opus/volltexte/2005/314.

Kas, M. (1993). *Essays on Boolean Functions and Negative Polarity*. Ph. D. thesis, University of Groningen.

Katz, G. (2000). Anti neo-Davidsonianism: Against a Davidsonian semantics for state sentences. In C. Tenny and J. Pustejovsky (Eds.), *Events as Grammatical Objects*, pp. 393–416. Stanford, California: CSLI Publications.

Keenan, E. and D. Westerståhl (1997). Generalized quantifiers in linguistics and logic. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of Logic and Language*, pp. 837–893. Cambridge, Massachusetts: MIT Press.

Keller, W. (1988). Nested Cooper storage: The proper treatment of quantification in ordinary noun phrases. In U. Reyle and C. Rohrer (Eds.), *Natural Language Parsing and Linguistic Theories*, pp. 432–447. Dordrecht: Reidel.

Kempson, R. and A. Cormack (1981). Ambiguity and quantification. *Linguistics and Philosophy 4*(2), 259–309.

Kim, J. (1998). Events as Property Exemplifications. In S. Laurence and C. Macdonald (Eds.), *Contemporary Readings in the Foundations of Metaphysics*, pp. 310–326. Blackwell Publishers.

Klima, E. (1964). Negation in English. In J. Fodor and J. Katz (Eds.), *The Structure of Language: Readings in the Philosophy of Language*, pp. 246–323. Prentice-Hall.

Koenig, J.-P. and G. Mauner (1999). A-definites and the discourse status of implicit arguments. *Journal of Semantics 16*(3), 207–236.

Koller, A. and S. Thater (2006). Towards a redundancy elimination algorithm for underspecified descriptions. In J. Bos and A. Koller (Eds.), *Inference in Computational Semantics ICoS-5 Proceedings*, Buxton, England, pp. 37–46.

Kratzer, A. (1995). Stage-level and individual-level predicates. In G. Carlson and F. Pelletier (Eds.), *The Generic Book*, pp. 125–175. University of Chicago Press.

Kratzer, A. (forthcoming). *The Event Argument*. Cambridge, Massachusetts: MIT Press.

Kroch, A. (1979). *The Semantics of Scope in English*. Garland Press.

Kuropka, D. (2004). *Modelle zur Repräsentation natürlichsprachlicher Dokumente: Ontologie-basiertes Information-Filtering und -Retrieval mit relationalen Datenbanken*. Number 10 in Advances in Information Systems and Management Science. Berlin: Logos Verlag.

Kurtzman, H. and M. MacDonald (1993). Resolution of Quantifier Scope Ambiguities. *Cognition 48*(3), 243–79.

Laenzlinger, C. (1998). *Comparative Studies in Word Order Variation: Adverbs, Pronouns, and Clause Structure in Romance and Germanic*. Amsterdam: John Benjamins Publishing Company.

Lakoff, G. (1971). On generative semantics. In D. Steinberg and L. Jakobovits (Eds.), *Semantics*, pp. 232–296. Cambridge. Massachusetts: Cambridge University Press.

Landauer, T., P. Foltz, and D. Laham (1998). An introduction to Latent Semantic Analysis. *Discourse Processes 25*(2-3), 259–284.

Lapata, M. (2000). The automatic interpretation of nominalizations. In *Proceedings of AAAI*, pp. 716–721.

Lapata, M. (2002). The disambiguation of nominalizations. *Computational Linguistics 28*(3), 357–388.

Lappin, S. (2000). An intensional parametric semantics for vague quantifiers. *Linguistics and Philosophy 23*(6), 599–620.

Lappin, S. and H. Leass (1994). An Algorithm for Pronominal Anaphora Resolution. *Computational Linguistics 20*(4), 535–561.

Levin, B. and M. Rappaport Hovav (1999). Two structures for compositionally derived events. In *Proceedings of SALT IX*, pp. 127–144. CLC Publications, Cornell University.

Light, M. and L. Schubert (1997). Knowledge representation for lexical semantics: is standard first order logic enough? In *Proceedings of the Second International Workshop on Computational Semantics IWCS-2*, Tilburg University.

Macleod, C., R. Grishman, A. Meyers, L. Barrett, and R. Reeves (1998). NOMLEX: A lexicon of nominalizations. In *Proceedings of the Eighth International Congress of the European Association for Lexicography*, Liège, Belgium, pp. 187–193.

Maienborn, C. (2005). On the limits of the Davidsonian approach: The case of copula sentences. *Theoretical Linguistics 31*(3), 275–316.

Mann, W. and S. Thompson (1988). Rhetorical structure theory: Toward a functional theory of text. *Text 8*(3), 243–281.

Manning, C. (2006). Local textual inference: It's hard to circumscribe, but you know it when you see it, and NLP needs it. Available at http://nlp.stanford.edu/ manning/papers/LocalTextualInference.pdf.

Marcu, D. (1997). The rhetorical parsing of natural language texts. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, Morristown, New Jersey, pp. 96–103. Association for Computational Linguistics.

Maxwell, J. and R. Kaplan (1993). The interface between phrasal and functional constraints. *Computational Linguistics 19*(4), 571–590.

May, R. (1985). *Logical Form: Its Structure and Derivation*. Cambridge, Massachusetts: MIT Press.

Meyers, A., C. Macleod, R. Yangarber, R. Grishman, L. Barrett, and R. Reeves (1998). Using NOMLEX to produce nominalization patterns for information extraction. In *The Computational Treatment of Nominals (Coling-ACL98 workshop)*, Volume 2, Montreal, Canada, pp. 25–32.

Miller, G., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller (2004). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography 3*(4), 235–244.

Moltmann, F. (1992). Reciprocals and same/different: Towards a semantic analysis. *Linguistics and Philosophy 15*(4), 411–462.

Montague, R. (1973). The proper treatment of quantification in ordinary English. *Approaches to Natural Language 49*, 221–242.

Mooney, R. (2007). Learning for semantic parsing. *Lecture Notes in Computer Science 4394*, 311.

Moortgat, M. (1997). Categorial type logics. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of Logic and Language*, pp. 93–177. Elsevier.

Nairn, R., C. Condoravdi, and L. Karttunen (2006). Computing relative polarity for textual inference. In J. Bos and A. Koller (Eds.), *Inference in Computational Semantics ICoS-5 Workshop Proceedings*, Buxton, England, pp. 67–76.

Nap, H. H. (2008). *Stress in Senior Computer Interaction*. Ph. D. thesis, Technical University of Eindhoven.

Osswald, R. (2005). On result nominalization in German. In E. Maier, C. Bary, and J. Huitink (Eds.), *Sinn und Bedeutung*, Volume 9, pp. 256–270.

Overberg, R. (forthcoming). *Illness Stories on the Internet*. Ph. D. thesis, Leiden University.

Overberg, R., L. Alpay, J. Verhoef, and J. Zwetsloot-Schonk (2007). Illness stories on the internet: What do breast cancer patients want at the end of treatment? *Psychooncology 16*(10), 937–44.

Overberg, R., P. Toussaint, and B. Zwetsloot-Schonk (2006). Illness stories on the internet: Features of websites disclosing breast cancer patients' illness stories in the Dutch language. *Patient Education and Counseling 61*(3), 435–442.

Padó, S. (2007). *Cross-Lingual Annotation Projection Models for Role-Semantic Information*. Ph. D. thesis, Saarland University.

Palmer, M., D. Dahl, R. Schiffman, L. Hirschman, M. Linebarger, and J. Dowding (1986). Recovering implicit information. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, pp. 10–19. Association for Computational Linguistics.

Palmer, M., N. Xue, O. Babko-Malaya, J. Chen, and B. Snyder (2005). A parallel proposition bank II for Chinese and English. In *Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, Ann Arbor, pp. 61–67. Association for Computational Linguistics.

Parsons, T. (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. Massachusetts: MIT press.

Parsons, T. (2000). Underlying states and time travel. In J. Higginbotham, F. Pianesi, and A. Varzi (Eds.), *Speaking of Events*, pp. 81–93. Oxford, New York: Oxford University Press.

Penka, D. and H. Zeijlstra (2005). Negative indefinites in Dutch and German. Available at http://ling.auf.net/lingbuzz/000192.

Pereira, F. and S. Shieber (1987). *Prolog and Natural-Language Analysis*. Stanford, California: CSLI Publications.

Pietroski, P. (2006). *Events and Semantic Architecture*. Oxford University Press.

Pietroski, P. and H. Hornstein (2002). Does every Sentence Like This Contain a Scope Ambiguity. In W. Hinzen and H. Rott (Eds.), *Belief in Meaning: Essays at the Interface*. Frankfurt: Hansel-Hohenhausen.

Poß, M. (forthcoming). *Under Construction*. Ph. D. thesis, Leiden University.

Poß, M. and T. van der Wouden (2005). Extended Lexical Units in Dutch. In T. van der Wouden, M. Poß, H. Reckman, and C. Cremers (Eds.), *Computational Linguistics in the Netherlands 2004*, Utrecht, pp. 187–202. LOT.

Potts, C. (2000). When even no's neg is splitsville. Available at http://ling.ucsc.edu/Jorge/index.html.

Pradhan, S., H. Sun, W. Ward, J. Martin, and D. Jurafsky (2004). Parsing arguments of nominalizations in English and Chinese. In *Proceedings of HLT-NAACL*, Boston.

Pustejovsky, J. (1995). *The Generative Lexicon*. Cambridge, Massachusetts: MIT Press.

Pustejovsky, J., J. Castano, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz (2003). TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the Fifth International Workshop on Computational Semantics IWCS-5*. Kluwer Academic Publishers.

Reckman, H. and C. Cremers (2006). Concepts across categories. In J. Bos and A. Koller (Eds.), *Inference in Computational Semantics ICoS-5 Proceedings*, Buxton, England, pp. 97–106.

Reckman, H. and C. Cremers (2007). Deep parsing semantic interpretation of nominalizations and their (un)expressed arguments. *Leiden working papers in Linguistics 4*(1), 40–55.

Reichenbach, H. (1947). The tenses of verbs. In H. Reichenbach (Ed.), *Elements of Symbolic Logic*. New York: The Macmillan Company.

Reinhart, T. and E. Reuland (1993). Reflexivity. *Linguistic inquiry 24*(4), 657–720.

Rothstein, S. (1995). Adverbial quantification over events. *Natural Language Semantics 3*(1), 1–31.

Rullmann, H. (1995). Geen eenheid. *TABU Squibnummer 2*(4), 194–197.

Saba, W. and J. Corriveau (2001). Plausible Reasoning and the Resolution of Quantifier Scope Ambiguities. *Studia Logica 67*(2), 271–289.

Sag, I., T. Baldwin, F. Bond, A. Copestake, and D. Flickinger (2002). Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, Mexico City, Mexico, pp. 1–15.

Sag, I. and T. Wasow (1999). *Syntactic Theory: A Formal Introduction*. Stanford, California: CSLI Publications.

Schein, B. (2002). Events and the semantic content of thematic relations. In G. Preyer and G. Peter (Eds.), *Logical form, Language & Semantic content: On Contemporary Developments in the Philosophy of Language and Linguistics*, pp. 236–344. Oxford, New York: Oxford University Press.

Schuurman, I. and P. Monachesi (2006). A semantic annotation scheme for Dutch. In K. Sima'an, M. de Rijke, R. Scha, and R. van Son (Eds.), *Proceedings of Computational Linguistics in the Netherlands 2005*, pp. 67–82.

Seuren, P. (2006). The natural logic of language and cognition. *Pragmatics: A Quarterly Journal of the International Pragmatic Association 16*(1), 103–138.

Seuren, P. (forthcoming). *The Victorious Square: A Study of Natural Predicate Logic*.

Solstad, T. (2007). Arguments in nominalisations: A unified approach to postnominal PPs and genitives in German. Presentation at the workshop Nominalizations Across Languages, Slides available at http://web.uni-frankfurt.de/fb10/rathert/forschung/nominalizations.html.

Steedman, M. (1996). *Surface Structure and Interpretation*. Cambridge, Massachusetts: MIT Press.

Steels, L. and J. Bleys (2005). Planning what to say: Second order semantics for fluid construction grammars. In *Proceedings of CAEPIA'05*, Lecture Notes in AI, Berlin. Springer Verlag.

Steels, L. and F. Kaplan (2001). AIBO's first words. the social learning of language and meaning. *Evolution of Communication 4*(1), 3–32.

Stevenson, S., A. Fazly, and R. North (2004). Statistical measures of the semi-productivity of light verb constructions. In *Proceedings of the ACL-04 Workshop on Multiword Expressions: Integrating Processing*, pp. 1–8.

Tatu, M. and D. Moldovan (2007). COGEX at RTE3. In *Proceedings of the Workshop on Textual Entailment and Paraphrasing*, Prague, pp. 22–27. Association for Computational Linguistics.

Terada, A. and T. Tokunaga (2003). Corpus based method of transforming nominalized phrases into clauses for text mining application. *IEICE Transactions on Information and Systems 86*(9), 1736–1744.

Van Trijp, R. (2008). The emergence of semantic roles in fluid construction grammar. In A. D. Smith, K. Smith, and R. Ferrer i Cancho (Eds.), *The Evolution of Language. Proceedings of the 7th International Conference*, Singapore, pp. 346–353. World Scientific Publishing.

van de Woestijne, C. (1999). A formal characterisation of the Delilah system. Master's thesis, Leiden University.

Vanderwende, L., D. Coughlin, and B. Dolan (2005). What syntax can contribute in the entailment task. In I. Dagan, O. Glickman, and B. Magnini (Eds.), *Proceedings of the 1st. PASCAL Recognision Textual Entailment Challenge Workshop*, pp. 13–16.

VanLehn, K. (1978). Determining the scope of english quantifiers. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Verhagen, A. (2003). The Dutch Way. In A. Verhagen and J. van de Weijer (Eds.), *Usage-Based Approaches to Dutch*, Number 1 in Occasional Series, pp. 27–58. Utrecht: LOT.

Verkuyl, H. and J. van der Does (1991). The semantics of plural noun phrases. In J. van der Does and J. van Eyck (Eds.), *Generalized Quantifier Theory and Applications*, pp. 403–439. Amsterdam: NLLI.

Visser, W. (2005). Reflexieven en pronomina in Delilah. Bachelor's thesis, Leiden University.

Voorhees, E. (2007). Overview of TREC 2007. In E. Voorhees and L. Buckland (Eds.), *The Sixteenth Text REtrieval Conference Proceedings 2007*, Number 500-274 in Special Publication, Gaithersburg, Maryland, pp. 1–16. NIST.

Voorhees, E. and D. Harman (Eds.) (2005). *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press.

Vossen, P., K. Hofmann, M. de Rijke, E. Tjong Kim Sang, and K. Deschacht (2007). The Cornetto database: Architecture and user-scenarios. In M.-F. Moens, T. Tuytelaars, and A. de Vries (Eds.), *Proceedings of the 7th Dutch-Belgian information retrieval workshop DIR 2007*, University of Leuven, pp. 87–96.

Wahlster, W. (Ed.) (2000). *Verbmobil: Foundations of Speech-To-Speech Translation*. Springer.

Willis, A. (2007). NP coordination in underspecified scope representations. In J. Geertzen, E. Thijsse, H. Bunt, and A. Schiffrin (Eds.), *Proceedings of the Seventh International Workshop on Computational Semantics IWCS-7*, Tiburg, pp. 235–246.

Wolf, L., R. Overberg, P. Toussaint, E. Hoenkamp, and H. Reckman (2006). Design of the Narrator system: processing, storing and retrieving medical narrative data. *Journal of Integrated Design and Process Science 10*(4), 13–33.

Zaenen, A., L. Karttunen, and R. Crouch (2005). Local textual inference: can it be defined or circumscribed? In *Proceedings of the ACL 2005 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, Michigan, pp. 31–36. Association for Computational Linguistics.

Zwarts, F. (1981). Negatief polaire uitdrukkingen I. *GLOT 4*(1), 35–132.

Zwarts, F. (1986). *Categoriale Grammatica en Algebraïsche Semantiek*. Ph. D. thesis, University of Groningen.

Samenvatting in het Nederlands

Natuurlijke taal stelt mensen in staat complexe informatie over te dragen. Het overdragen van informatie speelt een belangrijke rol in technische en wetenschappelijke vooruitgang. Sinds de uitvinding van het schrift kan informatie ook in de vorm van taal worden bewaard voor later gebruik. Nu teksten ook in digitale vorm kunnen worden opgeslagen, kunnen ze sneller doorzocht worden.

Een tekst wordt in de computer opgeslagen als een serie tekens. De meeste zoekmechanismen zijn gebaseerd op het vergelijken van rijtjes lettertekens: waar komt het rijtje dat gezocht wordt voor in de tekst? De computer heeft geen toegang tot de informatie die in de vorm van tekst gecodeerd is. Informatie waarmee de computer complexere taken moet kunnen uitvoeren wordt daarom niet opgeslagen in de vorm van lopende tekst, maar in de strak georganiseerde vorm van een database. Het maken van zo'n database is veel werk en je moet van te voren precies bedenken wat je ermee wil.

Dit proefschrift draagt bij aan het onderzoek dat probeert het begrijpen van informatie in menselijke taal te simuleren op de computer. Voor toepassingen betekent dit dat de computer zich gedraagt alsof hij begrijpt wat er in een tekst staat. Een computerprogramma dat dat kan, kan gebruikers beter helpen bij het zoeken naar en verwerken van informatie die is opgeslagen als gewone tekst.

Internet zoekmachines hebben het vinden van informatie al een stuk makkelijker gemaakt, maar de gebruiker moet nog veel zelf doen. Als je de computer in mensentaal kon uitleggen wat je zoekt en hij gerichter kon zoeken, documenten kon vergelijken en de resultaten op een inzichtelijke manier kon presenteren zou dat veel tijd schelen. De sprekende computer in Star Trek is een tot de verbeelding sprekend voorbeeld.

Semantiek is een deelgebied van taalkunde dat de betekenis van woorden, zinnen en teksten bestudeert. Een uiting in een taal is een reeks klanken, of op papier een reeks tekens. Maar voor degenen die de taal kennen is het veel meer dan dat. Taal is een code. De spreker codeert in zijn uiting informatie en de luisteraar decodeert en haalt die informatie er weer uit. Semantici willen weten hoe dit werkt. Hierbij gaat het vooral om de eerste betekenislaag (wat wordt er beweerd?), en niet zozeer om diepere interpretatie (hoe is dit bedoeld?). Het raden van elkaars bedoelingen is veel breder dan taal. En de direct gecodeerde betekenis is voorlopig al ongrijpbaar genoeg. We weten bijvoorbeeld nog heel weinig over hoe en in wat voor vorm we informatie in onze hersenen opslaan en verwerken.

De strategie die binnen het hier beschreven onderzoeksproject gebruikt wordt om begrip van talige informatie te simuleren is gebaseerd op het 'vertalen' van zinnen naar logische formules, als een poging om de betekenis op een voor de computer hanteerbare manier weer te geven. Formele logica maakt gebruik van een nauwkeurig gedefinieerde notatie waarin informatie wordt opgeschreven en van een systeem van redeneerregels

die vertellen hoe je nieuwe informatie kunt afleiden uit wat je al weet. Daar is formele logica ook voor; om op een systematische en controleerbare manier te redeneren. Er zijn computerprogramma's die de redeneerstrategieën van sommige formele logica's kunnen uitvoeren.

Binnen de computationele taalkunde wordt eerste orde logica veel gebruikt voor het weergeven van betekenis van uitingen in natuurlijke taal. Een eenvoudige zin als *elke man fietst* wordt dan bijvoorbeeld weergegeven met de volgende logische vorm:

$$\forall x.\ man(x) \rightarrow fietst(x)$$

Hier staat zoiets als: 'voor elke x geldt: als x een man is dan fietst x'. De $\forall$ noemen we een kwantor. Als de computer nu dit soort formules ter beschikking heeft, waarin bijvoorbeeld staat dat elke man fietst en dat Bob een man is, dan kan hij afleiden, volgens de regels van de logica, dat Bob fietst. Dit noemen we automatisch redeneren.

Eerste orde logica is niet noodzakelijk het beste formalisme om de betekenis van zinnen en teksten weer te geven. Het is duidelijk dat er bij het vertalen informatie verloren gaat. In de theoretische semantiek worden bijvoorbeeld ook hogere orde logica's gebruikt, maar die zijn voor de computer niet meer goed hanteerbaar. In deze dissertatie wordt gewerkt aan een manier om betekenis weer te geven die zowel informatiever als hanteerbaarder is dan eerste orde logica.

Het programma dat hier gebruikt is om taal naar logica te vertalen heet Delilah. Delilah ontleedt Nederlandse zinnen met behulp van kennis van de Nederlandse grammatica. Het is een sterk lexicalistisch systeem. Dit betekent dat het ingebouwde woordenboek (een database) een belangrijke rol speelt. Bij elk woord in het woordenboek staat een heleboel informatie, bijvoorbeeld wat de woordsoort is en met welke andere woordsoorten het een woordgroep vormt. Dit is dus eigenlijk grammaticale informatie, belangrijk voor de combinatoriek. Op deze manier wordt informatie *over* taal opgeslagen in een database om informatie *in* taal te kunnen verwerken. Elk woord brengt ook z'n eigen stukje logische vorm mee. In zo'n stukje logische vorm is steeds aangegeven waar de gaten zitten in de formule die moeten worden ingevuld door stukjes logische vorm van andere woorden in de grammaticale structuur (het stap voor stap samenvoegen van woorden tot woordgroepen en woordgroepen tot zinnen levert een hiërarchische structuur op). Zo worden de formules opgebouwd. De betekenisweergave van een zin is het resultaat van de stukjes betekenis die de woorden meebrengen en de manier waarop deze gecombineerd worden, wat bepaald wordt door de grammaticale structuur van de zin. Er wordt dus vanuit gegaan dat betekenis tenminste tot op zekere hoogte op een systematische manier berekenbaar is. Er zijn natuurlijk allerlei complicerende factoren. Zinnen zijn bijvoorbeeld vaak dubbelzinnig. Er is dan voor elke lezing een aparte vertaling nodig. Voor mensen is meestal meteen duidelijk wat de bedoelde interpretatie is, maar die gebruiken daarvoor een heleboel achtergrondkennis die moeilijk in een computer te stoppen is. Verder zijn er een heleboel woorden, zoals *hij* of *dat* die terugverwijzen naar een antecedent. Dit antecedent moet dan worden geïdentificeerd en de formule aangepast. De stijl van analyse die Delilah hanteert wordt wel 'diepe' analyse genoemd; de grammaticale structuur van zinnen wordt zo volledig mogelijk geanalyseerd en de betekenisanalyse is daarop gebaseerd. Deze nauwkeurigheid gaat typisch ten koste van de robuustheid en snelheid van het systeem.

Het Narrator project in het NWO programma voor Toegankelijkheid en Kennisontsluiting in Nederland, waarin het hier beschreven onderzoek was ingebed, streeft ernaar een systeem voor een website te ontwikkelen waarin borstkankerpatiënten kunnen zoeken naar voor hen relevante ervaringsverhalen van lotgenoten. Borstkankerpatiënten hebben vooral in het stadium na de behandelingen vaak behoefte aan lotgenotencontact. Dit kan helpen bij de emotionele verwerking van hun ervaringen. Ze lezen dan bijvoorbeeld op internet verhalen van anderen. Het internet heeft als voordelen voor de patiënten dat het laagdrempelig en anoniem is en dat er in principe veel verschillende verhalen beschikbaar zijn. Het zou daarbij ideaal zijn als er een geschikt zoekprogramma was dat iemand kon helpen om uit een verzameling verhalen over ongeveer hetzelfde onderwerp de meest relevante bij elkaar te zoeken om te lezen, bijvoorbeeld verhalen die goed aansluiten bij de eigen ervaringen. Het bieden van deze hulp bij het zoeken is de taak voor het Narrator systeem. Hoofdstuk één van dit proefschrift gaat over dit systeem, zoektechnieken en automatisch redeneren. De bedoeling is om een systeem te maken dat passende verhalen vindt door middel van gesimuleerd taalbegrip. Er kan dan meer dan bij het traditionele zoeken op 'keywords' het geval is, worden gezocht op inhoud.

Hoofdstuk twee beschrijft Delilah, bespreekt een aantal problemen en stelt oplossingen voor. Een interessant probleem is bijvoorbeeld de interpretatie van bijwoorden in bepaalde posities. In de zin *Waar denk je dat hij Alice gezien heeft?* vraagt *waar* bijvoorbeeld (in de meest voor de hand liggende lezing) naar de plaats van het *zien* (uit de bijzin). Bij de manier waarop bijwoorden tot nu toe geregeld waren in de grammatica was dit problematisch: *Waar* kon alleen geïnterpreteerd worden als vragend naar de plaats van het *denken* (uit de hoofdzin). Dit heeft te maken met het feit dat bijwoorden normaal gesproken niet verplicht zijn in de grammatica. *Ik denk dat hij Alice gezien heeft*, zonder plaatsbepaling in de bijzin, is bijvoorbeeld een prima zin. Als we dit vergelijken met een zin waar naar het lijdend voorwerp uit de bijzin gevraagd wordt, bijvoorbeeld *Wie denk je dat hij daar gezien heeft?*, zien we dat het hier wel meteen duidelijk is dat het vooropgeplaatste *wie* eigenlijk bij de bijzin hoort. Zonder lijdend voorwerp is de bijzin namelijk niet compleet. *Ik denk dat hij daar gezien heeft* is geen goede zin. De grammatica zit zo in elkaar dat als twee woorden of woordgroepen samengaan er altijd één is die de ander selecteert ('nodig heeft'). De selecterende partij noemen we het hoofd, de geselecteerde partij het argument. Een hoofd heeft een lijst van één of meer argumenten waarmee gecombineerd moet worden om een complete woordgroep of zin te vormen. Als een werkwoord en een lijdend voorwerp combineren tot een geheel, dan is het werkwoord het hoofd, want een transitief werkwoord kan alleen gebruikt worden in combinatie met een lijdend voorwerp. Bij het combineren van een werkwoord en een bijwoord, ligt het om verschillende redenen voor de hand om het bijwoord het hoofd te laten zijn. Dit lijdt echter tot het hierboven beschreven probleem. Dit probleem is opgelost door de verhoudingen in de grammatica om te draaien en het bijwoord een optioneel argument te laten zijn bij het werkwoord. Daardoor kan *waar* in ons eerste voorbeeld geïnterpreteerd worden als horend bij de bijzin.

Hoofdstuk drie gaat over een systematische verrijking van de logische vorm met informatie over de beschreven gebeurtenissen of 'events'. De betekenis van *elke man fietst* wordt dan als volgt weergeven:

$$\forall x.\ man(x) \rightarrow \exists e.\ gebeurtenis(e) \ \&\ fietsen(e) \ \&\ fietser/doener(e,x)$$

Hier staat zoiets als dat er voor elke man een gebeurtenis is die omschreven kan worden als fietsen en waarbij hij betrokken is als fietser; degene die het doet. Deze notatie maakt de gebeurtenis direct adresseerbaar. Er kan nu heel makkelijk informatie over de gebeurtenis worden toegevoegd, zoals de tijd en de plaats ervan. Daarvan wordt de formule wel langer, maar niet complexer. De informatie kan op hetzelfde niveau worden toegevoegd als de informatie over de betrokkene, die er al is. In de notatie die we eerder hebben gezien zou dat soort informatie de hele formule moeten inbedden en dus complexiteit toevoegen. De representaties zijn dus informatiever geworden en in sommige gevallen eenvoudiger qua structuur.

Werkwoorden zijn typisch de woorden die gebeurtenissen introduceren. Daarom zijn alle soorten werkwoorden zo aangepast dat er in het stukje formule dat ze bijdragen aan de zinsbetekenis verwezen wordt naar een gebeurtenis (of een toestand). Er zijn ook zelfstandige naamwoorden die naar een gebeurtenis verwijzen. Vaak zijn deze zelfstandige naamwoorden afgeleid van werkwoorden. We noemen dit nominalisaties. Een voorbeeld van een nominalisatie is het woord *operatie*, afgeleid van *opereren*. Als we nu naar de volgende twee zinnen kijken: *Alice werd geopereerd* en *Alice onderging een operatie* dan zien we dat beide gaan over eenzelfde soort gebeurtenis en dat Alice beide keren dezelfde rol speelt bij die gebeurtenis (die van patiënt). Het is zelfs zo dat de ene zin niet waar kan zijn zonder dat de andere ook waar is en andersom. Als er verschillen zijn, zijn dit misschien verschillen in stijl, of in de manier waarop de informatie wordt gepresenteerd, maar wat er beweerd wordt is in beide gevallen hetzelfde. Met de betekenisweergave nieuwe stijl wordt dit gemakkelijk herkend. Dus als een gebruiker zoekt naar teksten over mensen die geopereerd zijn, dan wordt een tekst waar één van de bovenstaande zinnen in staat gevonden. Het maakt niet uit welke formulering gekozen is.

Iets soortgelijks is gedaan voor bijvoeglijke naamwoorden (bijv. *boos*) en de daaraan verwante abstracte zelfstandige naamwoorden (bijv. *boosheid*). Beide refereren aan iets wat we een toestand of 'state' noemen. Dit leidt ook tot een nieuwe analyse van constructies van het type *honger hebben*. Deze constructie werd eerst beschouwd als een vaste verbinding. In de nieuwe aanpak heeft *hebben* eenzelfde soort functie als *ondergaan* in het *operatie*-voorbeeld en komt het belangrijkste deel van de betekenis van *honger*. Dit ondervangt de variatie die mogelijk is in deze constructie beter.

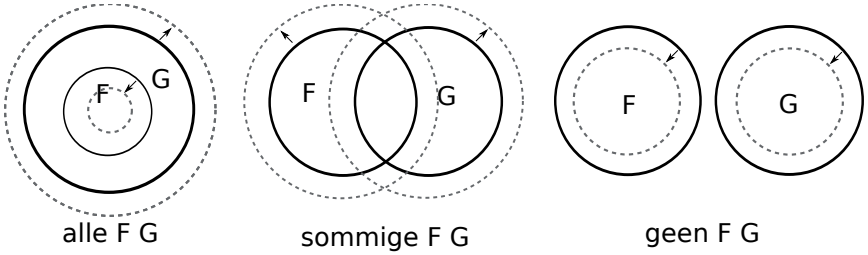
Een belangrijk concept in de semantiek is dat van logische consequentie. Om te weten of zin B een logische consequentie is van zin A, stel je je voor dat zin A waar is en ga je na of zin B in dat geval ook waar moet zijn. Als dat zo is, is zin B een logische consequentie van zin A en anders niet. De *opereren/operatie* zinnen hierboven zijn bijvoorbeeld logische consequenties van elkaar. Ons begrip van de betekenis van zinnen stelt ons in staat logische consequenties te herkennen. "B is een logische consequentie van A" zullen we schijven als $A \Rightarrow B$ (Uit A volgt logischerwijs B.) Als B geen logische consequentie is van A, schrijven we $A \not\Rightarrow B$.

Determinatoren zoals *alle*, *sommige* en *geen* spelen een belangrijke rol bij logische consequenties. Vergelijk bijvoorbeeld*:

*De voorbeelden met *geen* klinken misschien wat onnatuurlijk. Vervang dit patroon rustig door *geen aap*

alle apen geeuwen $\Rightarrow$ *alle mannetjesapen*$^{\downarrow}$ *geeuwen*
$\not\Rightarrow$ *alle apen geeuwen luidruchtig*$^{\downarrow}$
alle mannetjesapen geeuwen luidruchtig $\not\Rightarrow$ *alle apen*$^{\uparrow}$ *geeuwen luidruchtig*
$\Rightarrow$ *alle mannetjesapen geeuwen*$^{\uparrow}$
sommige apen geeuwen $\not\Rightarrow$ *sommige mannetjesapen*$^{\downarrow}$ *geeuwen*
$\not\Rightarrow$ *sommige apen geeuwen luidruchtig*$^{\downarrow}$
sommige mannetjesapen geeuwen luidruchtig $\Rightarrow$ *sommige apen*$^{\uparrow}$ *geeuwen*$^{\uparrow}$
geen apen geeuwen $\Rightarrow$ *geen mannetjesapen*$^{\downarrow}$ *geeuwen luidruchtig*$^{\downarrow}$
geen mannetjesapen geeuwen luidruchtig $\not\Rightarrow$ *geen apen*$^{\uparrow}$ *geeuwen luidruchtig*
$\not\Rightarrow$ *geen mannetjesapen geeuwen*$^{\uparrow}$

Je ziet dat wat uit wat volgt, afhangt van de determinator. Als uit iets specifieks iets algemeens volgt, noemen we dat stijgend. Als uit iets algemeens iets specifieks volgt noemen we dat dalend. *Alle* is dalend met betrekking tot z'n eerste argument (*apen*) en stijgend met betrekking tot z'n tweede argument (*geeuwen*), *sommige* is stijgend met betrekking to beide argumenten en *geen* is dalend met betrekking tot beide argumenten. In de voorbeelden is dit aangegeven met pijltjes naar boven en naar beneden. In figuur 1 wordt geïllustreerd hoe dit komt.



Figuur 1 — Venn-diagrammen die de determinatoren *alle*, *sommige*, en *geen* illustreren. De twee argumenten (bijvoorbeeld apen en geeuwers) zijn weergegeven als de verzamelingen F en G. De determinator vertelt wat de overlap moet zijn tussen de twee verzamelingen. Bij *sommige* mag die overlap bijvoorbeeld niet leeg zijn. Dit blijft zo als je van één of beide verzamelingen een bovenverzameling neemt. De pijltjes en cirkels in stippellijnen geven de stijgende en dalende richtingen voor logische consequenties aan.

In eerste orde logica kunnen deze logische consequenties berekend worden, tenminste als ergens is vastgelegd dat mannetjesapen apen zijn en apen dieren, enzovoorts. Om dat soort kennis op te zoeken bestaan ontologieën. Logica's zijn er op gemaakt dat je met weinig regels veel kunt afleiden. De redeneringen kunnen dan wel erg ingewikkeld worden. Eerste orde redeneringen zitten aan de bovengrens van wat een computer tegenwoordig kan. Als zo'n programma grotere hoeveelheden tekst langs moet om te kijken of wat je wil weten ergens uit volgt, gaat het al snel lang duren. Maar er is nog een

geeuwt, *er geeuwen geen apen* of *geen van de apen geeuwt*. De logische consequenties blijven hetzelfde. De zinnen zijn hier parallel gehouden met de andere, om de aandacht te vestigen op de verschillen die er in dit verband toe doen.

probleem. Natuurlijke talen hebben veel meer verschillende determinatoren dan eerste orde logica kwantoren heeft. De betekenis van *de meeste* kan bijvoorbeeld niet in eerste orde logica worden uitgedrukt.

De titel van dit proefschrift luidt: *Flat but not shallow*; 'Plat maar niet oppervlakkig'. Dit gaat over de eigenschappen die semantische representaties liefst zouden moeten hebben. Ze moeten enerzijds zo plat mogelijk zijn, in de zin dat ze zo min mogelijk hiërarchische structuur bevatten, omdat die het verwerken ervan bemoeilijkt. Daarom is er een trend ontstaan om te werken met *shallow semantics*. Het probleem met formalismen voor shallow semantics is dat je er nog minder in kunt uitdrukken. Je verliest dus extra informatie over de betekenis van de zin die je ontleedt. Typisch wordt de betekenis van determinatoren genegeerd.

In hoofdstuk vier wordt een nieuw soort logische vorm ontwikkeld, waarin de informatie behouden blijft maar de structuur versimpeld wordt. De representaties zijn eenvoudiger te verwerken omdat er al een heleboel voorwerk is gedaan voor het berekenen van logische consequenties. De betekenis van onze zin *elke man fietst* wordt nu als volgt weergegeven in Flat Logical Form (Platte logische vorm of FLF):

man(x+↓+elke+[]) &
gebeurtenis(e+↑+een+[]) &
fietsen(e+↑+een+[]) &
fietser/doener(e+↑+een+[],x+↑+elke+[])

De predicaten (*man*, *fietsen*...) zijn hetzelfde als in de vorige notatie en ook de variabelen (*x* en *e*) zien we terug. De variabelen zijn hier uitgebreid met extra informatie. De kwantoren zijn hier in weggewerkt. Ook is steeds lokaal (door middel van de pijltjes) de richting aangegeven die voor de logische consequenties van belang is. Er is ook nog een positie, die hier steeds wordt opgevuld door een lege lijst (*[]*), waar informatie over afhankelijkheden kan worden weergegeven. Was de zin bijvoorbeeld geweest *ik denk dat elke man fietst*, dan mocht daar niet zomaar uit worden afgeleid dat elke man fietst omdat deze informatie onder *denken* is ingebed. Om te kunnen voorkomen dat dit wel gebeurt moet deze afhankelijkheid worden vastgelegd en daar biedt die lijst op de laatste positie van de variabele ruimte voor. Nu volstaan eenvoudige redeneerregels om logische consequenties te berekenen.

FLF heeft als bijkomend voordeel dat ook determinatoren die geen vertaling hebben in de eerste orde logica meegenomen kunnen worden en dat je niet perse vast zit aan de redeneerregels van eerste orde logica, die niet altijd overeenkomen met de intuïties over natuurlijke taal.

Bij een zoektaak zoals die in Narrator kunnen nu zowel de zoekvraag (query) die de gebruiker ingeeft (of een afgeleide daarvan) en de teksten waarin gezocht moet worden in de vorm van FLF worden weergegeven. Door de representaties te vergelijken en enkele regels toe te passen kan dan gekeken worden waar de gezochte informatie in de tekst staat. Doordat een niveau van betekenis representatie gebruikt wordt, zit je niet vast aan de toevallige letterlijke formuleringen. Doordat FLF gebruikt is in plaats van eerste orde logica is het zoek- en redeneerproces relatief eenvoudig.

Als dit type onderzoek op de juiste weg is, is de Star Trek computer weer een klein stapje dichterbij.

Curriculum Vitae

Hilletje Gezina Bouwke (Hilke) Reckman was born in Groningen on the 26th of December 1978, and grew up in Appingedam, where she also attended the Ommelander College (1991-1997). After secondary school she spent a year in Verona, Italy as an au pair and learned Italian. This turned out to be an excellent way to gather the courage needed to make a decision on what to study. In 1998 she enrolled to study African Linguistics at Leiden University, in retrospect the most linguistically oriented first year program she could find. After the first year she continued in the General Linguistics program, from which she graduated cum laude in 2003. From 2003 till 2008 she held a position as a PhD student at the Leiden University Centre for Linguistics, participating in an NWO-funded interdisciplinary project. Her research during that period, on the Delilah parser and its semantic representations, resulted in the present study. At present she works as a lecturer at the Gerard Tuning Institute at Leiden University, teaching courses in computational linguistics and semantics.