Precise Understanding of Natural Language

Structural Semantics and Inference for Natural Language Texts

In Precisely-Formalizable Applications

An Introduction to my Dissertation's Research Direction

Iddo Lev

October 23, 2006

This document explains the overall research direction of my dissertation. Because this direction is different from most research today in mainstream NLP, I spend a considerable amount of space explaining it. The material here will be used as part of the dissertation's introductory part, and this document can be seen as an expanded dissertation proposal.

The dissertation will explore how structural-semantic representations that precisely capture the meaning of a Natural Language (NL) text can be calculated efficiently from the text, and how these representations can be used to intelligently infer conclusions that rely on an integration of information conveyed *throughout* the text in the context of precisely-formalizable applications. I see this dissertation as an important step in a broader research program which I call "Precise Understanding of Language by Computers" (PULC). For more information, please see that project's website.[1] All of my papers mentioned below are available on that website.

Section 1 explains what precise understanding of natural language is. Section 2 discusses what is needed to achieve that, and in particular what is structural semantics. Section 3 provides motivation for the research direction. Section 4 gives an outline of the dissertation content.

[1] http://www.stanford.edu/∼iddolev/pulc

1 Precise Understanding of Natural Language

1.1 What is Precise Understanding?

Creating computer programs that understand Natural Language (NL) texts is a notoriously difficult endeavor. The difficulty stems from the nature of NL: It is inherently imprecise and vague in ways that are hard to formalize; it it riddled with ambiguities that need to be resolved; the meaning of NL texts inherently depends on their context of use and the intention of the writer; and NL texts do not specify explicitly all the information that is intended to be conveyed since the writer can rely on the fact that the reader shares basic assumptions and world knowledge. In light of these difficulties, the question arises: What are useful scientific and technological approaches to making progress in the research on NL understanding and processing.

One approach, which is the prevalent one in mainstream NLP today, is to investigate any useful task that a computer can perform with NL input by using whatever means that are available. Simple examples include keyword-based search on texts, and more advanced examples include translation, summarization, information extraction, and question-answering. The aim is to create applications that are useful for human users even though they do not rely on deep understanding of the texts, and even though they cannot be precisely evaluated as to whether they achieve the task. Thus, for example, most machine translation programs do not even attempt to use explicit meaning representations that capture the information conveyed by the text, and they rely instead on statistical methods and word-alignment techniques (Brown et al., 1993). Judgments about the quality of the output of such programs may vary significantly between humans. Nonetheless, these programs can be considered successful because they produce approximate solutions that are judged by many people to be good enough for certain purposes (such as a first-pass rough translation aiding a human translator).

The research program described here takes a different approach. It does not deny that NL is inherently imprecise and that for principled reasons even humans cannot in general precisely understand every NL text. Nevertheless, the approach does point out that there are precisely-formalizable applications that employ NL inputs, as we shall see below. In those applications, precise understanding of NL is in fact possible thanks to the nature of the application, and the research direction here is to investigate how to achieve such understanding.

By "precisely-formalizable application" I mean a computer program that implements a mathematically, rigorously-defined or otherwise precisely-defined procedure for solving some problem. Most computer applications are in fact of this kind. The clearest examples are programs that implement well-understood procedures for solving math or physics problems, such as solving certain differential equations or calculating the trajectory of missiles under idealized conditions. Other examples include compilers of programming languages, which are constructed based on a precise written specification of the compiler's output given various input conditions, and database systems that use query languages such as SQL. Most "mundane" programs such as word processors are also based on precise specification documents which detail how the program should behave under various conditions (e.g. if the user selects

the "undo" function then the most recently performed action from a certain class should be reversed). Such an application could be given a mathematical definition (e.g. a graph representing the structure of menus or actions) even if this is not spelled out explicitly in practice. I will use the term "precise application" as a shorthand for these kinds of precisely-formalizable applications. They contrast with applications such as computer graphics, where it is seldom possible to give a precise specification of the desired picture or animation output, and evaluation relies on human judgement of whether and to what extent the rendered pictures look real.

Thus, the research direction here is concerned with precise applications, except that the input is not given in the usual formalized manner but instead in a NL text that describes the precise input. Although in the general case of NL understanding there is no guarantee that the NL input can be precisely formalized or that there is a precise answer that all humans could agree upon, these things are guaranteed in the context of precise applications because the humans generating the NL input have a specific end-formalization in mind. The main research challenge is how to bridge over the difference between the inherent impreciseness of NL and the precise end-formalization. Although the term "precise understanding (of natural language)" is not completely accurate because we are not promising here precise understanding in *any* NLP application, it is useful as a shorthand for precise applications that deal with NL.

1.2 Precise Understanding Applications

Given the inherent impreciseness of NL that was mentioned above, it might at first seem strange to talk about precise applications that deal with NL. But there are in fact such applications.

1.2.1 Controlled Language

A controlled language is a restricted subset of NL which is so precisely delineated that it effectively becomes a formal language which just looks (almost) like NL.[2] Such languages have been used in manuals and other technical texts in various areas of industry (such as the aero-space industry[3]) and in other specification tasks (see e.g. (Nelken and Francez, 1996) and (Fuchs et al., 2006)[4]). The motivations for these applications is that a controlled NL increases naturalness and ease of learning for human users compared to a formal specification language while still maintaining complete precision and predictability. Here is an example text written in Attempto Controlled English, specifying the operation of an ATM (taken from (Fuchs, 2005)):

(1) Every customer has at least 2 cards and their associated codes. If a customer C approaches an automatic teller and she inserts her own card that is valid carefully into the slot and types the correct code of the card then the automatic teller accepts the card and displays a message "Card accepted" and C is happy. No card that does not

[2]See e.g. http://www.ics.mq.edu.au/∼rolfs/controlled-natural-languages/.
[3]See http://www.boeing.com/phantom/sechecker/se.html.
[4]See also http://www.ifi.unizh.ch/attempto/.

have a correct code is accepted. It is not the case that a customer's card is valid, and is expired or is cancelled.

This text is quite complex and sounds almost completely natural. The ACE engine automatically translates it and similar texts to correct first-order logic (FOL) representations of the specifications conveyed by the texts. One way to demonstrate correctness is to show that NL sentences (in the controlled language) which humans judge to follow from the text are in fact translated to FOL formulas that logically follow from the text's representation, whereas sentences that are judged not to follow are translated to formulas that do not logically follow from the text.

If NL is inherently imprecise, how is the computer able to accurately understand such texts? This is possible thanks to two things: First, the human who wrote this text had a precise FOL end-formalization in mind. Second, the allowed NL input is restricted to avoid imprecision and to be disconnected from particular contexts. Moreover, potential ambiguities are usually handled in such systems by fiat: When the computer encounters a NL construction which is ambiguous in general, a specific one of its possible interpretations is always selected, and the definition of the controlled language declares in advance which one it is. For example, PP attachment ambiguity is always resolved by attaching the PP to the preceding NP rather than the VP. If the user wants to convey the alternative choice, she needs to use a paraphrase (e.g. putting the phrases in a different order or using an explicit relative clause), or to use some artificial means such as a comma. This practice makes the NL input at times a little less than completely natural, but it is an effective compromise.

The research direction I want to pursue shares with controlled languages the goal of precisely understanding the user input, but I do not want to place any a-priori restrictions directly on the possible range of input NL constructs or what they mean. Instead, the restriction is placed on the application as a whole – it should be precisely-formalizable. This overall restriction ensures that, by definition, what each input sentence means and what should be done with it under difference circumstances is completely clear *when the context of the application is taken into account*. This restriction may indirectly lead to some restrictions on the range of allowed NL phenomena in the input, but this need not be the case.

Despite the differences, research on controlled languages still aims to gradually expand the range of linguistic constructions the system can handle. In that respect, the work of this dissertation can directly feed into that endeavor.

1.2.2 Word Problems

There are many problems in physics and mathematics for which there are well-understood and mathematically-precise procedures. These are so precisely defined that it is possible to write computer programs that carry out these calculations correctly. The input to these programs usually needs to be expressed in a formalized language.

Students who are taught these procedures are given practice problems to improve and test their skills of using these procedures. The practice problems are usually given not in a formalized way but using a NL description of the problem. The writers of the text start from a formalized problem and then convert it to an NL

Preamble: Six sculptures – C, D, E, F, G, and H – are to be exhibited in rooms 1, 2, and 3 of an art gallery. The exhibition must conform to the following conditions:

(1) Sculptures C and E may not be exhibited in the same room.
(2) Sculptures D and G must be exhibited in the same room.
(3) If sculptures E and F are exhibited in the same room, no other sculpture may be exhibited in that room.
(4) At least one sculpture must be exhibited in each room, and no more than three sculptures may be exhibited in any room.

Question 1: If sculpture D is exhibited in room 3 and sculptures E and F are exhibited in room 1, which of the following may be true?

(A) Sculpture C is exhibited in room 1.
(B) No more than 2 sculptures are exhibited in room 3.
(C) Sculptures F and H are exhibited in the same room.
(D) Three sculptures are exhibited in room 2.
(E) Sculpture G is exhibited in room 2.

Question 2: If sculpture G is exhibited in room 1, which of the following may NOT be a complete list of the sculpture(s) exhibited in room 2?

(A) Sculpture C (B) Sculptures E and H (C)...

Adapted from Weber (1999).

Figure 1: Example of a logic puzzle text

description. There is usually more than one NL text that can describe the same formalized problem. The student's task is to understand the NL description and rediscover the formalized problem, and then apply the appropriate procedure to solve it. When the student reads the NL text, she relies on the assumption that there is a precise end-formalization from which the NL text was created.

Given this assumption, solving physics, chemistry, math, or logic problems from their NL descriptions is a central example of precise understanding applications. Even if the NL input text contains parts that, taken out of context, pose serious difficulties for automated understanding because they are imprecise, ambiguous, and depend on a complex psychological model of the writer's intentions, all these issues can be dealt with when the text is understood within the context of the word problem solving application.

In this dissertation I will mainly focus on an application that solves logic puzzles of the kind appearing in LSAT and GRE exams. An example puzzle is shown in Figure 1. In contrast to other word problems which may require complex knowledge of the subject matter, solving logic puzzles relies on very simple procedures (essentially, finite constraint-satisfaction). Thus, we can focus our research efforts on the NL understanding part without also having to worry about a difficult back-end computation. Furthermore, while solving physics problems requires access to a rich ontology of concepts and a related lexicon, logic puzzles generally rely on very little lexical and world knowledge. The major research challenge is that logic puzzles require an accurate understanding and merging of the logical constraints expressed *throughout* the text – even a small misunderstanding usually leads to completely wrong answers. For a proposal to create a test-suite of logic puzzle texts, see (Lev, 2006d). For preliminary work on a system for solving logic puzzles, see (Lev et al.,

2004).

If the domain knowledge required for solving a class of word problems can be precisely formalized, the knowledge and methods developed for solving logic puzzles would be useful in a NL front-end that translates the word problem to formalized representations. A good example is math questions such as those on SAT exams (solving such puzzles given their English description was one of the tasks that was suggested in discussions of the next DARPA Grand Challenge.) Recently, a knowledge-based system developed in project HALO[5] was able to get a reasonably high score on a chemistry AP test after the text was encoded in a knowledge representation language. A natural next step to work on is to try giving the computer the ability to read and understand the NL text directly and translate it to the knowledge representations.

1.2.3 Computational Law

A real-world application that has some similarity to solving logic puzzles is understanding regulation texts, such as a website that describes which collection of courses a college student must take in order to fulfil the requirements of a study program, as in (2) below.[6] As with logic puzzles, regulation texts describe general rules and conditions that must be met. The computer should be able to answer questions about these rules as well as questions based on an additional description of a particular real or hypothetical situation (e.g. check whether the set of courses a student has taken conforms to the regulations). Also similarly to logic puzzles, answers to such questions rarely if ever appear explicitly in the text, and must be *inferred* from it. Not all legal texts can be formalized precisely (some are written to be deliberately vague), but many can, and they would be good candidates for precise understanding.

(2) A candidate is required to complete a program of 45 units. At least 36 of these must be graded units, passed with an average 3.0 (B) grade point average (GPA) or higher. The 45 units may include no more than 21 units of courses from those listed below in Requirements 1 and 2.[7]

1.2.4 NL Interface to Databases and Knowledge Bases

Instead of querying a database using a formal language like SQL, it would be very useful for users if the computer could understand NL questions and commands such as:

(3) a. Which department has the largest number of employees?
b. How many employees were paid a salary higher than $70,000 over the last two years?
c. Update: John Smith works in the personnel department.

There are two approaches to designing a NL interface to a database (NLIDB) (see (Androutsopoulos et al., 1995) for a survey). One approach is to direct the user to ask any question on a certain topic, and then try to use several existing databases

[5] http://www.projecthalo.com/

[6] For a general motivation for the field of computational law, see http://complaw.stanford.edu.

[7] From: http://cs.stanford.edu/Degrees/mscs/degree.php.

that may have relevant information to answer the question. In this case, there is no guarantee that the conceptual understanding of the domain that the user has in mind aligns well with the conceptual view that exists implicitly in the design of the databases.

Another approach from the point of view of a precise application is to instruct the user that the interface is essentially a database interface language such as SQL, except that it is expressed in NL (see e.g. (Nelken and Francez, 2000; Androutsopoulos, 2002)). In particular, the user should only expect the system to understand very straightforward requests, just as those possible in SQL. The only thing the user should not need to worry about in advance are potential NL ambiguities because they should be resolvable either through interaction with the user or automatically based on the data in the databases and assumptions about the context of the application (as with any other precise understanding application).

To demonstrate this, consider the question "Did at least two supervisors attend every repair session?". This question has a potential scope ambiguity between the two readings:

(4) a. $\textit{at-least}[2](\textit{supervisor}, \lambda x.\textit{every}(\textit{repair-session}, \lambda y.\textit{attend}(x, y)))$
b. $\textit{every}(\textit{repair-session}, \lambda y.\textit{at-least}[2](\textit{supervisor}, \lambda x.\textit{attend}(x, y)))$

However, if the application is supplied with domain knowledge that says it is impossible for one supervisor to attend all repair sessions, then the only relevant reading is (4)b. If in fact the answer to that query is *true* in a given database, the system could answer "yes" even though the answer to the query (4)a is *false* in that database. If such knowledge is unavailable and the system cannot disambiguate the question, it should say so to the user. In effect, this is treating NL as a formal language except that in some contexts, an expression may contain an actual ambiguity and the system has to report this (just as a compiler could report an unresolvable ambiguity of a polymorphic operator in a particular context of use).

A similar application but more complex is NL interfaces to knowledge bases (KBs). There are many KBs that were designed to model a particular domain such as intelligence analysis [REFERENCE]. Existing interfaces to such KBs involve rigid forms, which can only address simple queries, or complex formal languages, which require expertise on behalf of the user. It would be helpful to connect the formalized knowledge with a NL front-end. This is more complex than NLIDBs because chains of inference may be involved. As with the precise understanding view of NLIDBs above, the user should be instructed not to expect any more than the formal interface provides.

Now that we have seen what precise understanding is and that it is feasible in principle, two questions arise: Why is it worthwhile to pursue this direction? and: How can it be achieved? To answer the first question, we first need to discuss the second.

2 What is Needed for Precise Understanding

2.1 The Main Research Questions

The following is a list of research questions that need to be answered when precise understanding applications are created.

1. What **semantic representation language** should we use to express the meaning of NL texts? The representations should have a well-defined interpretation which allows us to understand them and predict what inferences they support (see e.g. (McDermott, 1978) for more on this point).

2. What do particular NL sentences **mean**, and how should this meaning be expressed using the semantic representations? The answer is sometimes not clear if a sentence is taken out of context (e.g. how many readings does the sentence "four boys lifted five pianos" have?).

3. How can the representations be **calculated** from NL texts? To answer this, we need to know how the morphology and syntax of the input texts can be calculated. We also need to know what are the algorithms and knowledge that comprise the **syntax-semantics interface**, which specifies how the semantic representations are calculated from the syntactic analysis of the input NL sentence.

4. How should **NL ambiguities** be handled? This includes: how should the representations be calculated efficiently without an exponential explosion, and how should the sentences be disambiguated?

5. How can the representations **support inference**?

6. NL texts do not supply explicitly all the domain or world knowledge as they assume the reader has this knowledge. Are there ways to circumvent this **lack of knowledge** without supplying all of it by hand?

It is beyond the scope of one dissertation to answer or even address all these questions. Hence, this dissertation will address them primarily from the point of view of developing an application that can solve logic puzzles, while at the same time keeping an eye on the general questions, and testing the developed knowledge and algorithms on a few examples from other applications. The main emphasis will be put on the development of the needed structural semantics knowledge and relevant algorithms for the syntax-semantics interface. To make it clear, I next define what I mean by structural semantics.

2.2 What Is Structural Semantics?

Roughly speaking, the branch of linguistics called *structural semantics*[8] deals with the literal meanings of sentences after these meanings are abstracted away from the

[8]The terms *formal semantics* or *compositional semantics* are commonly used instead. I prefer the term *structural semantics* as it distinguishes this branch of semantics from lexical semantics, which may also be formalized, and which also addresses issues of composition, e.g. noun-noun compounds or the alteration in meaning that verbs and nouns undergo when they are combined in a metonymy.

concepts that non-functional words are related to. It is similar to the semantics of a logical language, where what the symbols in the vocabulary stand for is irrelevant for defining the semantics of the language abstractly, and the only thing that matters is the syntax of the language and the meaning of the logical connectives. Structural semantics explains how the structural meaning of a sentence is related to the syntactic structure of the sentence and the meaning of functional units (the equivalent of logical connectives in a formal language), which include morphological markers, quantifiers, determiners, logical connectives, modals, auxiliary words, pronouns and relative pronouns, some prepositions, and ellipsis markers.

As a simple illustration, the inference in (5), which is based only on structural semantics, does not require knowing the meaning of the words *John*, *pencheon*, and *sopomous*. It only requires knowing that they are a proper name, a noun, and an adjective, respectively, as well as the meaning of the functional words *every*, *is*, and *a*. Additionally, one needs to have an accurate knowledge of morphology and syntax as prerequisites for the structural semantics stage.

(5) Every pencheon is sopomous.
John is a pencheon.
⇒ John is sopomous.

In contrast, the inference in (6)a is not *merely* a structural semantic inference because it depends on the meaning of the words *man* and *human* and on world knowledge about the connection between those meanings. However, if this knowledge is given to us in explicit form, as in (6)b for instance, then the pattern in (6)a can still provide a test for structural semantic knowledge.[9]

(6) a. Every human likes watermelons.
⇒ Every man likes watermelons.

b. $\forall x.[man(x) \rightarrow human(x)]$

The inference in (7)a and the difference between (7)a and (7)b depend not only on the meaning of the words but also on linguistic knowledge about argument realization – how thematic roles are connected to syntactic roles. This knowledge is intimately connected to issues of knowledge representation and the ontology, it is complex and very wide in scope, and is not considered part of structural semantics.

(7) a. John loaded the wagon with hay.
⇒ The wagon became full [at that time].

b. John loaded hay on the wagon.
$\not\Rightarrow$ The wagon became full [at that time].

Finally, structural semantics does not encompass world knowledge of the kind necessary for drawing the inference in (8).

(8) John ate a steak in a restaurant.
⇒ John probably talked to a waiter before he ate the steak.

[9] An NL sentence expressing background knowledge may itself introduce ambiguities and require more world knowledge in order to understand it. Since the goal here is to test only the understanding of (6)a, the background knowledge in (6)b is formalized rather than given as an NL sentence.

3 Motivation for the Research Direction

3.1 Why Precise Understanding?

Why is precise understanding of NL a useful endeavor? There are two perspectives.

The scientific perspective is that no system exists today that can read an unseen word problem text in a certain domain and solve it correctly, and it would be interesting to investigate how to construct such a system. If the computer did manage to understand these complex texts and correctly solve the problems they describe, that would necessarily mean that we have discovered something non-trivial about NL, and have learned things that have implications on linguistics and psychology.

The technological perspective is that, as we have seen above, there are in fact useful real world applications that can be classified under precise understanding or a closely related rubric, and the techniques developed for the research direction here would be immediately applicable. The practical motivation for precise understanding is that not compromising precision for breadth of coverage is essential for system reliability, which is required for specification languages and certain NL interfaces.

Additionally, what the research here could teach us, especially about computational structural semantics, would be useful also for other kinds of NLP applications both in the short and the long term, as will be explained below.

3.2 Why Structural Semantics?

Whereas it is quite well-understood how to calculate the morphological and syntactic structures of NL texts, this is not the case with semantics. There has been some progress in the research on computing semantic representations from syntactic structures, e.g. (Alshawi, 1992) and more recent research on underspecified representations (Blackburn and Bos, 2005a,b; Egg et al., 2001; Copestake et al., 2005). Yet much work remains to be done on making the algorithms efficient for dealing with ambiguities, making the formalism convenient for human knowledge writers, and extending the coverage of analysis to more advanced linguistic constructions.

It is particularly important to work on extending this coverage. As the examples in section 1.2 show, the computer needs to know about functional semantic constructions, such as quantifiers, numbers and plurals, modifications, connectives, comparative and superlative constructions, conditionals, modals, anaphora, and more. As we saw, it also needs to know about potential scope ambiguities. Without further progress on these, we would be able to feed the back-end inference component only quite simple semantic representations. Therefore, the natural place to concentrate our efforts is on the research on computational structural semantics.

Working on structural semantics knowledge would also have an impact on NLP applications beyond precise understanding. Such knowledge is of course insufficient *by itself* for most inferences from NL texts as they depend heavily on lexical semantics and on domain and general world knowledge. Nonetheless, structural semantics is a necessary component in any system that does interesting inference based on the *meaning* of NL input and the *information* conveyed by texts. Without this knowledge, one is limited to just the meaning of words and only very rudimentary combinations of them.

Furthermore, this knowledge is largely domain-independent, and so it possesses a level of generality higher than other kinds of knowledge. Thus, once it is developed, it would have a large impact on many NLP applications, and it could be used for different purposes with little customization (the main customization might be adding frequencies of various phenomena, which may differ across domains). The next section surveys some of these applications.

Finally, of all the kinds of semantic knowledge needed in a sophisticated NL understanding system (including lexical knowledge and world-knowledge in ontology and facts), the body of structural semantic knowledge has the smallest size, and so we have a good chance to capture all or almost all of it through a concentrated collaborative effort in a reasonable amount of time (of course this project extends much beyond the scope of one dissertation).

3.3 Other NLP Applications

3.3.1 Question Answering

The matching between queries and texts that existing question-answering systems (such as (Pasca and Harabagiu, 2001)) calculate is based mainly on word alterations and matching syntactic structures. The quality of this matching would be improved if it also relied on knowledge of structural semantics. This knowledge would be used to help capture and represent more precisely the meaning and information that are actually conveyed by the texts and to perform higher-level reasoning. While arguably such knowledge may not be needed for answering simple who-did-what-to-whom factoid questions whose answers appear explicitly in the text, it is necessary for being able to answer questions that rely on *combining* the information that is conveyed by several sentences. For example:

(9) T: [Some NL texts talking about U.S. presidents, e.g.:]
George Walker Bush (born July 6, 1946) is the 43rd President of the United States, inaugurated on January 20, 2001. He was re-elected in 2004 and is currently serving his second term.[10]
Q: Was the third president of the United States to be re-elected a Democrat or a Republican?

It is unlikely that the answer appears explicitly in some text as "The third president of the United States who got re-elected was a Democrat" or some variation of it. Rather, the computer must understand what *third* means and how it combines with the meaning of the VP complement to constrain the choice of president, as well as understand the meaning of the connective *or*. It also has to calculate the answer by combining various pieces of information conveyed throughout the given texts.

In the RTE challenge (Dagan et al., 2005),[11] the computer needs to decide whether a hypothesis sentence can be inferred from a given short text. Knowledge of structural semantics would be useful for many RTE-like questions. For example, in (10), the computer needs to know what *more than* means (the answer would be *Does not follow*

[10] From http://en.wikipedia.org/wiki/George_W._Bush.
[11] See also http://www.pascal-network.org/Challenges/RTE/Introduction/.

if *80 kilometers* were replaced with *120 kilometers*). In (11), the computer needs to know how to instantiate a statement quantified by *each* (here, with the year 2005), as well as to know about implications between numeric quantifiers. Answering (12) correctly requires understanding conditionals and modality.

(10) T: In any case, the fact that this week Michael Melvill, a 63-year-old civilian pilot, guided a tiny rocket-ship more than 100 kilometres above the Earth and then glided it safely back to Earth, is a cause for celebration.[12]
H: A tiny rocket-ship was guided more than 80 kilometers above the Earth. [Follows]

(11) T: Each year, 26,000 people are killed or mutilated by landmines of which 8,000 are children.
H: In 2005, at least two thousand children were injured by landmines. [Follows]

(12) T: Things would be different if Microsoft was located in Georgia.
H: Microsoft's corporate headquarters are not located in Georgia. [Follows]

NLP systems that go beyond syntactic and lexical knowledge to actually knowing what functional words mean and how they tie together different pieces of information would enjoy a competitive edge over similar systems that do not use such knowledge.

3.3.2 Knowledge Acquisition from NL Texts

The more structural language knowledge a computer has, semantics included, the better able it is to automatically acquire factual knowledge correctly from crawling texts on the internet. Possessing only knowledge of syntax allows only rudimentary acquisition of simple patterns of facts that appear explicitly in the text; but having more sophisticated semantic representations and inference can allow the computer to combine separate pieces of information that appear throughout the texts. This is precisely the utility of semantic representations, that they capture the *content* of a text, and make it possible to relate the pieces of information to each other, merge them, compute entailments between them, etc. These are not possible if one relies only on the *form*, i.e. syntax, of the texts.

3.3.3 Annotation and Pre-Annotation Research

An important future goal is the creation of a Semantic Treebank, i.e. a corpus similar to the Penn Treebank except sentences would be annotated not only with syntactic structures but also with semantic representations of their meanings. This would support progress in statistical semantics of the kind that the Penn Treebank supported in statistical syntax.

The creation of the Penn Treebank, and especially its 300-page annotation manual that essentially codified a syntactic grammar, was possible thanks to a more-or-less broad consensus that existed in the field regarding what syntactic representations should be used. This consensus was reached only after several decades of research on syntax in theoretical linguistics and on rule-based parsers in NLP. A similar kind

[12] From the first RTE test-suite.

of progress in computational structural semantics is a necessary prerequisite for a Semantic Treebank. However, the levels of consensus, understanding, and coverage in structural semantics in linguistics and computational linguistics today are less than they were for syntax in the 1990s. Many issues still remain unresolved, and more progress on them is needed before good annotation schemes can be developed. In fact, even in syntax, there is still much room for improvement in the Penn Treebank, for example in the internal structure of noun phrases, and regarding ellipsis. Working on precise understanding provides the basic research in computational structural semantics which could eventually lead to the creation of a Semantic Treebank.

4 Outline of the Dissertation

The dissertation will consist of four parts: 1) Introduction and background, 2) Computational syntax-semantics interface, 3) Linguistic analysis, 4) Inference, application, and testing.

4.1 Introduction and Background

The first part will discuss the goals and approach of the dissertation. Because the approach is different from contemporary mainstream NLP, this part will put an emphasis on motivation and justification, extending what I wrote in the preceding sections.

4.2 Computational Syntax-Semantics Interface

This part explains how semantic forms can be efficiently computed from a syntactic analysis. The tools developed will be used in the rest of the dissertation.

4.2.1 Syntax-Semantics Interface

The traditional approach to specifying the (structural) syntax-semantics interface, starting with Montague's work (Montague, 1974) (see also e.g. (de Swart, 1998)) adheres to a very simple scheme: The meaning of a syntactic constituent is calculated by using only lambda-application to combine the meanings of the constituent's immediate sub-parts. This idea works for simple sentences but runs into difficulties when quantifiers and other more complex linguistic constructions are involved. Consequently, the rules mapping syntax to semantics become complicated, e.g. using type-raising operators.

In contrast, Glue Semantics (Dalrymple, 2001) provides a more powerful and flexible scheme of combination. While allowed combinations are restricted by the syntactic structures, the order of their computation need not necessarily adhere to the hierarchical structure of the syntax. This allows for a simpler specification of the syntax-semantics connection. The price is that the scheme of combination is more complex – it does not rely on a simple lambda-application principle but on a more powerful logical proof system (implicational linear logic). However, the price is worth paying because it needs to be paid only once, and the framework makes it easier

for the linguist or grammar engineer to specify the syntax-semantics connection. For more information, please see the introduction to glue semantics I wrote: (Lev, 2006c).

A computational implementation of glue semantics has been created by the NLTT group at the Palo Alto Research Center during the 1990s.[13] This was made into a component of the XLE system (XLE is a large computational infrastructure for analyzing and generating NL texts, which has been developed by the NLTT group since the early 1980's[14]). The component takes as input syntactic analyses in LFG (Bresnan, 2001; Dalrymple, 2001), and by following glue semantics specifications it creates semantic representations. The implementation is based on the algorithm of Hepple (1996) with some optimizations. Some additional experimental implementations were based on Proof Nets or on the ideas in (Gupta and Lamping, 1998).

4.2.2 Handling NL Ambiguities

Hepple's algorithm and the implementation at PARC were not constructed to deal efficiently with the prevalent ambiguities in NL. For each possible syntactic analysis, the algorithm runs a separate calculation, and each possible calculation yields a separate semantic representation. Representations multiply also as a result of scope ambiguities of quantifiers and other operators. This is inefficient and the number of results could easily explode.

The XLE's philosophy of handling NL ambiguities says that no component has complete information locally to choose the correct analysis and completely disambiguate the input. Instead, all possible analyses are represented and calculated efficiently in a packed form with the help of a *choice space* (Maxwell and Kaplan, 1991). This is similar to the idea underlying a classic $O(n^3)$ chart parser, whose chart forest output compactly represents an exponential number of analyses, and each analysis can be retrieved in linear time from the forest (see e.g. (Allen, 1995)). In the XLE, this idea is generalized to other levels of linguistic analysis: Morphology (Kaplan et al., 2004b), F-structure (Maxwell and Kaplan, 1993, 1996), and knowledge representation (Crouch, 2005). The XLE also uses statistical methods to filter out some of the locally less-likely analyses in each component (Kaplan et al., 2004a).

The glue semantics implementation has not been previously integrated with the choice space framework. One of the main aims of this dissertation is to do just that. The goal output should be a packed semantic representation. For example, the two possible semantic representations for "Bill saw the girl with the telescope" are shown on the left of Figure 2. Since they have a lot of material in common, we would like to obtain the packed representation on the right instead of the two representations in explicit form. The solid arrows correspond to the interpretation where the girl is with the telescope, while the broken arrows correspond to the other interpretation. Notice that parts that are shares between the two semantic structures appear just once, and most of the differences are just the pointers.

This packed structure bears some similarity to underspecified semantic representa-

[13]The people who participated in this included: Ash Asudeh, Dick Crouch, Mary Dalrymple, John Fry, Vineet Gupta, Angie Hinrichs, John Lamping, Jonathan Reichenthal, Vijay Saraswat, and Martin van der Berg.

[14]http://www2.parc.com/isl/groups/nltt/xle/

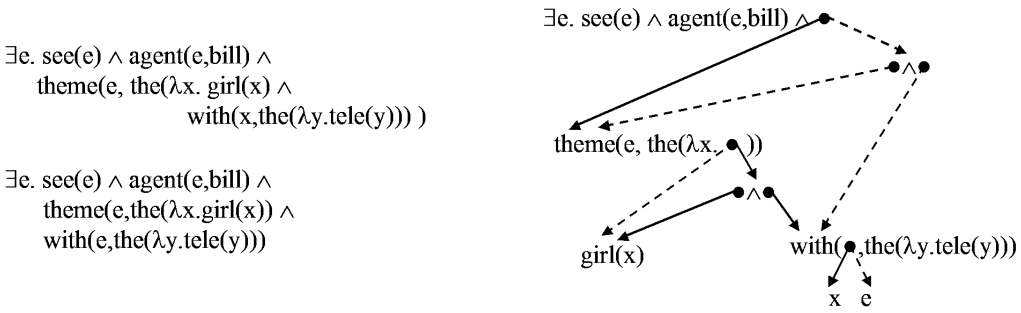

Figure 2: Packed semantic representation

tions (Bos, 1996; Egg et al., 2001; Copestake et al., 2005). While these methods share the goal of compactly representing competing semantic representations, they were developed in quite ad-hoc ways by complicating the syntax-semantics specification with extra overhead to manage ambiguities and pieces of underspecified structures. Instead, the goal here is to have the packed semantic representations fall out naturally from the interaction between glue semantics and the choice-space framework. Just as with the packed rewriting system in the XLE (Crouch, 2005), the grammar writer should be allowed to think (almost) only about non-ambiguous inputs, leaving it to the choice-space mechanism to *automatically* take care of managing the ambiguities and keeping representations in a packed form as in Figure 2.[15]

During my internship at PARC in summer 2006 under the supervision of Dick Crouch, I wrote a new program that combines glue semantics with the choice-space mechanism. I am currently documenting this work. Briefly, the main idea is that we want to know all the possible ways of creating a glue category[16] before we move on with the rest of the derivation. This is achieved by first computing a category graph, where each subcategory appearing on the right-hand-side of some glue statement gets a node, and edges connect the nodes X and $X \to Y$ to Y. The strongly-connected components of this directed graph are then found, and a topological sort on the induced DAG is calculated. The resulting node order determines the order in which we need to calculate the packed glue derivation.

For this work, I also wrote a core glue lexicon which specifies how pieces of F-structure should be translated to glue semantics statements. What remains to be done in this area is to extend this glue lexicon to additional linguistic constructions, and to complete the integration of my program with the XLE.

4.3 Linguistic Analysis

In this part, I will extend the coverage of the glue lexicon to more advanced constructions that have been given little to no attention in the computational semantics literature. The particular selection of topics will be motivated by the NL construc-

[15] A related proto-attempt to achieve this goal is reported in (Lev, 2005b), but it uses quasi-logical forms (Alshawi, 1992) and hole semantics (Bos, 1996) rather than glue semantics.

[16] Recall from (Lev, 2006c) that a glue category appears on the right-hand-side of a glue semantics statement, such as $saw : \mathbf{1}^e \to \mathbf{2}^e \to \mathbf{3}^t$, and atomic categories roughly correspond to places in the syntactic structure, e.g. $\mathbf{1}^e$ and $\mathbf{2}^e$ correspond to the NP nodes, and $\mathbf{3}^t$ to the sentence node.

tions that are most common in logic puzzle texts. I have already produced an extensive analysis in glue semantics of comparatives (Lev, 2005c,a), overt reciprocals (*each other*) and covert ones (as in *same* and *different*) (Lev and Garcia, 2005; Lev, 2006e). I have some preliminary work on anaphora (Lev, 2006a) and plurals (in draft form). I have also analyzed modals and conditionals (Lev, 2004), as well as events (Lev, 2006b,f). Please see these papers for more information.

The overall guiding methodology in this kind of endeavor is an iterative development process together with a systematic and careful linguistic analysis of NL phenomena. We start with existing analyses in (Dalrymple, 2001), drawing from additional standard sources such as (Carpenter, 1998; Blackburn and Bos, 2005a,b), and then iteratively expand the coverage. Most of the work will involve extending the glue lexicon, but some work is also needed in extending the coverage of the syntactic grammar of the XLE, in particular for comparatives and ellipsis.

The process consists of systematically investigating one NL phenomenon after another, consequently refining and extending the linguistic knowledge. The investigation of NL phenomena consists of looking at sentences that we want the computer to be able to understand and figuring out what these sentences mean. We could ask this question generally, i.e. what are all the possible meanings of the sentences in all possible contexts, but we also focus this question by looking at particular contexts in logic puzzles because in those contexts, it is very clear what the sentences mean.

For each meaning, we try to write it down using the semantic representation language. If it is possible, we try to figure out how to break this meaning down to its components that come from words and syntactic constructions in a way that is consistent with the existing syntax-semantics interface. If either is not possible, we need to revise the semantic representation language, the syntax-semantics interface, or some other part of the system to accommodate the new data. This revision might include changing previously accumulated knowledge and algorithms. After these revisions, we need to run regression tests to make sure that none of the capabilities of the system before the change were adversely affected.

In order to ensure that the results are as general as possible, we need to look at many different variations of our sentences of interest, so that we do not end up developing an analysis which is too specific to a few instances. We can be guided by relevant research in linguistics so that we do not end up re-inventing the wheel. The goal is to integrate all this knowledge into one consistent and coherent whole. We need to keep in mind good grammar engineering principles which separate general behaviors we agree on from specific implementation details that might very well change in later revisions. Of course, I will not necessarily be able to provide complete solutions to the topics because there are some complex cases that are hard to deal with. Luckily, they are quite rare in NL texts and also do not appear in logic puzzles.

4.4 Inference, Applications, and Testing

The final part of the dissertation will show how the semantic representations that are produced as described in the previous parts of the dissertation can be used to support inferences and help solve correctly a few logic puzzles. For a preliminary system for solving logic puzzles that I worked on in 2004, see (Lev et al., 2004).

4.4.1 Inference

The result of the packed derivation in glue semantics is a packed semantic representation, as in Figure 2. How can we do inference on it? Ideally, the idea of compactly representing all possible analyses should be carried forward to the automated reasoning stage. This has been partially done at PARC in the system they built for the RTE challenge. The meaning of a text is represented as a set of skolemized facts (derived from the semantic representations), relativized to various contexts in the choice-space if there is an ambiguity in the input. Then, a hypothesis is said to be entailed from a text if its facts match a subset of the facts in the text (unless negation is involved) (Crouch, 2005; Bobrow et al., 2005; Nairn et al., 2006).

However, this is a specialized kind of inference, and the more general theorem provers and model builders that are available today do not currently support packed inference. Extending them to do so is a very interesting goal, but it is beyond the scope of this dissertation. Therefore, we currently need to unpack the output of the glue semantics derivations.

After the semantic representations are unpacked, we want to reason with them by using off-the-shelf automated reasoners (theorem provers and model builders). However, these only accept input in first-order logic (FOL). FOL itself is not a good choice for a semantic representation language (SRL) for NL because the *shape* of the formulas is quite different from the surface NL structure, making it hard to calculate them compositionally from the syntactic analysis of the NL input. However, FOL can still serve as the "implementation level" for inference if we translate the semantic representations to FOL. The translation should preserve the representations' meaning under the assumption that we use a set AX of additional FOL axioms that define the meaning of operators used in our SRL, such as set membership. In other words, we want to get: $\Gamma \vdash^{SRL} \varphi \quad \Leftrightarrow \quad tr(\Gamma) \cup AX \vdash^{FOL} tr(\varphi)$.

4.4.2 Intelligent Understanding

What happens if the NL input has an ambiguity? Like most research in NLP, we can use statistical preferences to first extract the locally most likely analysis out of the packed representation. Judging what is the locally most likely syntactic analysis is already done in the XLE. Judging what is the locally most likely scoping of operators can be done by using heuristics as in (Alshawi, 1992, ch.8) or by a classifier as in (Higgins and Sadock, 2003; Andrew and MacCartney, 2004).

However, doing *only* that would not utilize the full power that is available to us. The unique aspect of the NL-understanding application here is that the ultimate arbitrator regarding which interpretation is correct is based on a global consideration of the *content* of the entire text in the particular *context* of application. The inference we can do using the precise semantic representations can tell us whether an analysis of one sentence is logically consistent with the analyses chosen for previous sentences, thus helping us choose the correct analysis, even if it is not the statistically most likely one locally. Checking that the chosen interpretation is informative, i.e. not entailed by previous sentences, can further reduce ambiguities. Finally, the context of the logic puzzles task provides a powerful final filter – our interpretation of the text must lead to exactly one correct answer for each multiple-choice question. It would be

particularly interesting to show specific examples where local considerations choose the wrong analysis, whereas the global inference-based consideration helps choose the right one.

A final issue to overcome is knowledge gaps in the text. Luckily, puzzle texts are quite explicit. Still, some gaps exist, e.g. to solve the puzzle in Figure 1, the computer needs to know that a sculpture may not be exhibited in more than one room, but this is not stated explicitly. Inference and the context of the puzzle can again help here: Knowledge gaps can often be recast as mathematical properties of relations. The unique location constraint on sculptures is equivalent to constraining the mapping from sculptures to rooms to be one-to-one; other cases exist of constraining mappings to be onto or total. Such properties can be obtained through a systematic search for sets of implicit constraints that, in combination with the explicitly stated information, yield exactly one answer per question.

4.4.3 Testing

I will test the entire system as described here on sentences from logic puzzle texts and on whole puzzles. For the first, the tests will check whether the representations entail desired conclusions. For the second, the tests will check whether the whole puzzle is under the linguistic coverage of the system, and whether the puzzle gets the correct solution.

Because of the nature of the system and the hand-crafted knowledge, the testing will not be broad scale but will target a few puzzles and sentences. In the context of a precise-understanding application where precision is the primary criterion and is not to be compromised for breadth of coverage, the main point is not so much to test the system's coverage (although that is important) as it is to investigate the reasons for the system's failure on unseen inputs, and to see how easily the system's coverage can be extended to cover these cases.

4.5 A Final Note

It is important to point out that the approach proposed here is quite different from classical work in NLP/AI.

First, systems in the 60s and 70s, such as (Bobrow, 1967; Winograd, 1972; Woods, 1973) were based on very ad-hoc semantics since linguistic semantics was very undeveloped at the time (Montague's seminal work (Montague, 1974) appeared only in 1974). In contrast, the research here draws from three decades of research in structural semantics.

Second, classical systems often suffered from a gross lack of modularity. For example, in SHRDLU and (Schank, 1975; Schank and Riesbeck, 1981), one component did both syntactic and semantic analysis at the same time in a way that was partly hard-coded rather than expressed cleanly in a declarative way. In fact, these works partly aimed to be psychological models of human understanding, and so this lack of modularity was seen as a virtue – it was assumed that in the human brain, all components interact together. However, from an engineering perspective, it is well known that non-modular systems become unwieldy very quickly, and a clean conceptual separation between modules makes the algorithms simpler to develop and maintain. In the

research here, modularity is pursued by separating components dealing with syntax, semantics, and reasoning, and by separating declarative knowledge from algorithms that use it.

Finally, classical systems were very domain-dependent: Properties specific to the particular task of the system influenced all the components. For example, anaphora resolution in SHRDLU was a complex hard-wired procedure that intimately depended on the particular structure of the dialogues and properties of the blocks world. Consequently, it was very hard to port such systems to other domains. In contrast, the semantic algorithms and knowledge developed here, as well as other processing in the XLE, do not depend on the domain of logic puzzles or any other specific domain or task. They are based on a general linguistic analysis of NL. It is only the back-end of the system which adds additional procedures and heuristics that adapt the general linguistic components to the task of logic puzzles. Even there, we aim as much as possible to find general principles that hold across many domains, e.g. consistency and informativity checking for a sentence given its predecessor sentences. Therefore, the algorithms and knowledge developed here are portable to many other NLP applications (they might need additions, but only little modification of existing code).

References

Allen, James. 1995. *Natural language understanding*. Benjamin Cummings.

Alshawi, Hiyan, ed. 1992. *The core language engine*. MIT Press.

Andrew, Galen, and Bill MacCartney. 2004. Statistical resolution of scope ambiguity in natural language. http://nlp.stanford.edu/nlkr/scoper.pdf.

Androutsopoulos, I., G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases–an introduction. *Journal of Language Engineering* 1:29–81.

Androutsopoulos, Ion. 2002. *Exploring time, tense, and aspect in natural language database interfaces*. John Benjamins Publishing Company.

Blackburn, Patrick, and Johan Bos. 2005a. *Representation and inference for natural language: A first course in computational semantics*. CSLI Publications.

Blackburn, Patrick, and Johan Bos. 2005b. *Working with discourse representation theory: An advanced course in computational semantics*. http://www.blackburnbos.org/.

Bobrow, Daniel G. 1967. Natural language input for a computer problem solving system. In *Semantic information processing*, ed. M. L. Minsky, 133–215. MIT Press.

Bobrow, Danny, Cleo Condoravdi, Richard Crouch, Ronald Kaplan, Lauri Kartunnen, Tracy Holloway King, Valeria de Paiva, and Annie Zaenen. 2005. A basic logic for textual inference. In *Proc. of the AAAI Workshop on Inference for Textual Question Answering*.

Bos, Johan. 1996. Predicate logic unplugged. In *Proc. of the 10th Amsterdam Colloquium*, 133–142.

Bresnan, Joan. 2001. Lexical Functional Syntax.

Brown, Peter F., Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematic of statistical machine translation: Parameter estimation. *Computational Linguistics* 19:263–311.

Carpenter, Bob. 1998. *Type-logical semantics*. MIT Press.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal Recursion Semantics: an introduction. *Research on Language and Computation* 3:281–332.

Crouch, Richard. 2005. Packed rewriting for mapping semantics to KR. In *Proc. of the 6th IWCS*.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proc. of the PASCAL Challenge Workshop on Recognizing Textual Entailment*.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics Series*. Academic Press.

Egg, Markus, Alexander Koller, and Joachim Niehren. 2001. The constraint language for lambda structures. *Journal of Logic, Language, and Information* 10:457–485.

Fuchs, Norbert E. 2005. Attempto Controlled English. Talk slides, Stanford University, December 2005. http://www.ifi.unizh.ch/attempto/talks/files/Talk.Stanford.05.pdf.

Fuchs, Norbert E., Kaarel Kaljurand, and Gerold Schneider. 2006. Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In *Proc. of FLAIRS'2006*.

Gupta, Vineet, and John Lamping. 1998. Efficient linear logic meaning assembly. In *Proc. of COLING/ACL'98*.

Hepple, Mark. 1996. A compilation-chart method for linear categorial deduction. In *Proc. of COLING'96*.

Higgins, Derrick, and Jerrold M. Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics* 29:73–96.

Kaplan, R., S. Riezler, T. H. King, J. T. Maxwell III, A. Vasserman, and R. Crouch. 2004a. Speed and accuracy in shallow and deep stochastic parsing. In *Proc. of HLT-NAACL'04*.

Kaplan, Ronald, John T. Maxwell III, Tracy Holloway King, and Richard Crouch. 2004b. Integrating finite-state technology with deep LFG grammars. In *Proc. of the ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP*.

Lev, Iddo. 2004. A basic analysis of conditionals. Paper for Ling233A, seminar on Conditionals, Stanford University.

Lev, Iddo. 2005a. Comparative constructions. Addendum to (Lev, 2005c).

Lev, Iddo. 2005b. Decoupling scope resolution from semantic composition. In *Proc. of the 6th International Workshop on Computational Semantics (IWCS-6)*, 139–150.

Lev, Iddo. 2005c. Gradable comparatives: Syntax and syntax-semantics interface. Paper for Ling221B, class on HPSG, Stanford University.

Lev, Iddo. 2006a. Anaphora in Glue Semantics.

Lev, Iddo. 2006b. Events in Glue Semantics.

Lev, Iddo. 2006c. Introduction to the syntax-semantics interface using Glue Semantics.

Lev, Iddo. 2006d. Logic puzzles: A new test-suite for compositional semantics and reasoning. *Submitted* .

Lev, Iddo. 2006e. On the syntax-semantics interface of overt and covert reciprocals. Paper for Ling223B, seminar on Quantification, Stanford University.

Lev, Iddo. 2006f. Some steps in formalizing events. Paper for Ling233, seminar on Tense and Aspect, Stanford University.

Lev, Iddo, and Ivan Garcia. 2005. On the syntax-semantics interface of *Different* and *Each Other*. Handout given at the 6th Semantics Fest, CSLI, Stanford University.

Lev, Iddo, Bill MacCartney, Christopher D. Manning, and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Proc. of the 2nd workshop on text meaning and interpretation, ACL'04*.

Maxwell, John T., III, and Ronald M. Kaplan. 1991. A method for disjunctive constraint satisfaction. In *Current issues in parsing technology*, ed. Masaru Tomita, 173–190. Kluwer Academic Publishers. Reprinted in Dalrymple et al. (eds), Formal Issues in Lexical-Functional Grammar, CSLI, 1995.

Maxwell, John T., III, and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics* 19:571–590.

Maxwell, John T., III, and Ronald M. Kaplan. 1996. Unification-based parsers that automatically take advantage of context freeness. In *Proc. of LFG'96 Conference*. http://www.parc.com/research/publications/details.php?id=3115.

McDermott, Drew. 1978. Tarskian Semantics, or No Notation Without Denotation! *Cognitive Science* 2.

Montague, R. 1974. The proper treatment of quantification in ordinary English. In *Formal philosophy*, ed. R. Thomason. Yale University Press.

Nairn, Rowan, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *In the Proc. of ICoS-5 (Inference in Computational Semantics)*.

Nelken, Rani, and Nissim Francez. 1996. Translating natural language system specifications into temporal logic via DRT. Technical report LCL 9602, Laboratory for Computational Linguistics, Technion.

Nelken, Rani, and Nissim Francez. 2000. Querying temporal databases using controlled natural language. In *Proc. of COLING'2000*.

Pasca, Marius, and Sanda M. Harabagiu. 2001. High performance question/answering. In *Proc. of SIGIR*, 366–374.

Schank, R. C. 1975. *Conceptual information processing*. North-Holland Publishing Co.

Schank, R. C., and C. K. Riesbeck. 1981. *Inside computer understanding: Five programs plus miniatures*. Lawrence Erlbaum Associates.

de Swart, Henriette. 1998. *Introduction to natural language semantics*. CSLI.

Weber, Karl. 1999. *The unofficial guide to the GRE test*. ARCO Publishing, 2000 edition.

Winograd, Terry. 1972. Understanding natural language. *Cognitive Psychology* 3:1–191.

Woods, W. A. 1973. Progress in natural language understanding: An application to lunar geology. In *AFIPS Conference Proceedings*, volume 42, 441–450.