Language Understanding by Reference Resolution in Episodic Memory

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Kevin Michael Livingston

EVANSTON, ILLINOIS

December 2009
This dissertation presents an approach to language understanding that treats all ambiguity resolution as a problem of reference resolution: grounding references to episodic memory. This model of language understanding is evaluated with an implementation of DMAP (Direct Memory Access Parsing) called REDMAP (Reference resolution in Episodic memory for DMAP). DMAP is a language understanding model that recognizes its input by mapping phrasal patterns to existing knowledge structures, updating memory with new information only as needed.

REDMAP works with a large logic based memory (evaluated with ResearchCyc 1.28 million assertions). It uses lexically driven rules to form candidate sets of assertions, and queries memory to ground references in those assertions to existing instances. Assertions from subsequent sentences are merged with running interpretations by identifying how new references are mapped to existing references. Mappings are evaluated by propagating remindings to existing instances to the new references. These instances are substituted into the new assertions, and memory is queried for their existence. If found these assertions support the reference mapping. Additionally, these queries will simultaneously ground any new unmapped references, if possible.
A corpus of simplified English texts describing people, places, and events that span multiple sentences and multiple texts was used to evaluate the accuracy and scalability of this approach. This dissertation provides strong support for two claims. Claim 1: A memory-based reference resolution algorithm (REDMAP) can provide broad coverage of and extending an existing large knowledge base by grounding to existing episodic memory as it parses and can use that memory to reduce ambiguity. Claim 2: The reading rate (mean time per sentence for a text) of the REDMAP algorithm is empirically independent the number of references in the text and the length of input (number of sentences). The evaluations also provide weaker support for an additional two claims. In contrast to supporting claim 1, the savings obtained by REDMAP are mitigated by the cost of additional interaction with memory; however the overhead is shown to be constant and minimal. Furthermore, REDMAP performs knowledge integration while resolving references, alleviating the need for this to be conducted as a separate step.
ACKNOWLEDGEMENTS

My continual education has had many teachers and supporters along the way, and several have played important roles with respect to my learning as a graduate student. Jennifer Seitzer encouraged me to apply to graduate school, and I am glad I listened to her advice. Larry Birbaum helped bring me to Northwestern, and Ken Forbus helped me stay; both provided valuable input on my thesis committee. I cannot thank Chris Riesbeck enough for all the time, energy, and patience he has put into being my thesis advisor. Chris guided me to become a better scientist and teacher, while providing a continual example as to how to be a better person. I enjoyed my time as Chris’s student, and I will miss being able to walk down the hall to his office to talk.

My fellow graduate students, past and present, are too numerous to list, but so many have helped me and been my friend through it all. Abhishek Sharma and Leo Ureel deserve special recognition for providing contributions and conversations throughout the Learning Reader project. Madeline Usher is a colleague who has provided ideas and code at critical moments, while simultaneously being a good friend.

Sometimes graduate school can seem a terminal condition, and several people have helped me continue to see the big picture, including Pat Langley, and Henry Lieberman. Larry Hunter provided the motivation to pull everything together at the end – a job. I look forward to continuing my education in Denver.
Dan and Lydia Halstead have always been there for me, and I could not have made it without them “adopting” me. Dan Dausch saw some of my best and worst times while we were roommates, and always made sure I had opportunities to take a break. The Adorni family has always been kind to me and made my time in Chicago more fun and interesting. There are many others who have helped me in Chicago and who have been my friends. The staff of Beck’s and the Loyola library deserve thanks for giving me the coffee and space necessary to write most of the early drafts of this dissertation, while always making me feel welcome.

My family has provided unwavering support throughout (even those who wondered if I would ever finish), and I appreciate their kindness and patience. My parents, Bob and Mary Beth Livingston, have always encouraged my education; their support has helped make it all possible. A special thank you is owed to those who remind me to take notice of the world around me: some, such as my youngest cousins, have shown me you are never too young to teach, and others, such as my great-great-aunt Dorothy (age 95), that you are never too old to learn. My sister, Erin, has gone through difficulties that put mine in perspective, while simultaneously accomplishing feats I am incapable of; she has provided inspiration and example throughout.
Table of Contents

Table of Contents ............................................................................................................................ 7
List of Figures ..................................................................................................................................... 10
List of Tables ................................................................................................................................... 12
Chapter 1  Introduction ................................................................................................................. 13
  1.1 Example .................................................................................................................................. 15
  1.2 Understanding and Reference ................................................................................................. 17
  1.3 Pipeline Parsing ...................................................................................................................... 18
  1.4 Memory-Based Parsing ........................................................................................................... 19
  1.5 Motivation ............................................................................................................................... 21
  1.6 Reference Resolution in Episodic Memory ............................................................................ 23
  1.7 Claims ..................................................................................................................................... 24
  1.8 Dissertation Overview ............................................................................................................ 26
Chapter 2  Knowledge Base and Corpus ...................................................................................... 28
  2.1 Knowledge Base ..................................................................................................................... 28
    2.1.1 Collections ....................................................................................................................... 29
    2.1.2 Relations .......................................................................................................................... 31
    2.1.3 Non-Atomic Terms (NAT) .............................................................................................. 33
    2.1.4 High-Order Relations ....................................................................................................... 34
    2.1.5 Consistency and Microtheories ...................................................................................... 34
  2.2 Corpus Description ................................................................................................................. 36
    2.2.1 Simplified English ........................................................................................................... 37
    2.2.2 Generating Simplified English ......................................................................................... 40
    2.2.3 Breakdown of Corpus Contents ....................................................................................... 41
Chapter 3  Mapping Text to Semantic and Episodic Structures ................................................... 44
  3.1 Lexicon ................................................................................................................................... 44
    3.2 Rules ....................................................................................................................................... 46
      3.2.1 Rule Phrasal Patterns ....................................................................................................... 47
      3.2.2 Rule Bases ....................................................................................................................... 54
      3.2.3 Nested Pattern Matching Example .................................................................................. 57
  3.3 Knowledge Extracted from ResearchCyc ............................................................................... 60
    3.3.1 Extracted Lexicon Content .............................................................................................. 61
      3.3.1.1 Proper Names ............................................................................................................ 61
      3.3.1.2 Lexical Concepts ....................................................................................................... 61
      3.3.1.3 Denotations to Semantic Concepts ........................................................................... 62
    3.3.2 Extracted Rules ................................................................................................................ 63
      3.3.2.1 Verbs ......................................................................................................................... 64
      3.3.2.2 Multi-word Nouns ..................................................................................................... 66
      3.3.2.3 Adjectives ................................................................................................................. 69
      3.3.2.4 Bad Rules .................................................................................................................. 70
Chapter 4  Reference Resolution .................................................................................................. 73
  4.1 Reference Resolution Example ............................................................................................... 74
4.2 Forming Understandings

4.2.1 Reference Resolution Algorithm

4.3 Examples of Leveraging Memory

4.3.1 Filtering by Shared Exemplars

4.3.2 Incremental Reference Resolution

4.3.3 Reference Resolution with Multiple Remindings

4.3.4 Conflicting Role Fillers

4.3.5 Semantic Expectations

4.4 Challenges in Interacting with Memory

4.4.1 Knowledge Organization

4.4.2 Engineering Limitations

4.5 Discussion

5.1 Knowledge Base Coverage

5.1.1 Discussion

5.2 Accuracy of Understanding

5.2.1 Assertion Accuracy

5.2.2 Interpretation Coherence

5.3 Partial Understandings and Interpretation Quality

5.3.1 Lexicon and Rule Coverage of Corpus

5.3.2 Effects of Partial Understanding

5.3.3 Discussion

5.4 Question Answering

5.5 Discussion

Chapter 5 Coverage and Quality

5.1.1 Discussion

5.2 Accuracy of Understanding

5.2.1 Assertion Accuracy

5.2.2 Interpretation Coherence

5.3 Partial Understandings and Interpretation Quality

5.3.1 Lexicon and Rule Coverage of Corpus

5.3.2 Effects of Partial Understanding

5.3.3 Discussion

5.4 Question Answering

5.5 Discussion

Chapter 6 Scalability: Ambiguity and Speed

6.1 Incremental Updates

6.1.1 Word Level Exploration (WLE)

6.1.2 Sentence Level Exploration (SLE)

6.1.3 Reuse Driven Exploration (RDE)

6.1.4 Scalability of Incremental Updates

6.2 Influences on Rate of Reading

6.2.1 Overall Reading Rate

6.2.2 Pattern Matching Time

6.2.3 Effect of Text Length

6.2.4 Effect of References

6.2.5 Discussion

6.3 Effects of Existing Knowledge

6.3.1 Input

6.3.2 Experiment

6.3.3 Results

6.3.4 Discussion

6.4 Impact of Existing Knowledge Structures

6.4.1 Existing Knowledge versus Ambiguity

6.4.2 Existing Knowledge versus Time
List of Figures

Figure 2.1 Subset of the ResearchCyc ontology near the collection Person. Arrows represent a generalization relationship between two collections. ................................................................. 30

Figure 2.2 Subset of the ResearchCyc ontology under the collection Event. Arrows represent a generalization relationship between two collections. ................................................................. 30

Figure 3.1 Diagram of T1 matching rule-s-Bomb-o, primary components bolded. Links are annotated with assertions from ResearchCyc that facilitated the connection............................... 52

Figure 3.2 Diagram of T2 matching rule-s-Bomb-o, primary components bolded. Links are annotated with assertions from ResearchCyc that facilitated the connection............................... 53

Figure 3.3 Diagram of T3 matching rule-s-Target-o and rule-Event-In-Loc, primary components bolded............................................................................................................................................ 60

Figure 5.1 Histogram of the percent of paragraphs (y axis) from the extended corpus producing the given number of assertion clusters (x axis). ........................................................................... 116

Figure 5.2 Histograms of the percent of paragraphs (y axis) in a corpus section producing the given number of assertion clusters (x axis), ranging from 0 to 7. ...................................................... 117

Figure 5.3 Histogram per corpus section of the ratio of words understood by REDMAP per text. ..................................................................................................................................................... 120

Figure 5.4 Histograms per corpus section of the ratio of words used by REDMAP rules per text ..................................................................................................................................................... 121

Figure 5.5 Quality of the assertions being produced by REDMAP vs. partial understanding threshold........................................................................................................................................... 123

Figure 5.6 Quantity of the assertions being produced by REDMAP vs. partial understanding threshold............................................................................................................................................... 124

Figure 6.1 Number of sentences (y axis) that were processed under a given time limit (x axis), for three different incremental update algorithms. The points correspond to increasing powers of 2 on the x-axis. ........................................................................................................................................... 135

Figure 6.2 Mean reading rate per sentence in milliseconds on the x-axis, vs. the percent of texts read at or under that rate on the y-axis. Vertical lines at 50%, 90%, 95%, and 99%. The x-axis is clipped at 10 seconds; there are two points missing, 11,636, and 18,773. .............................................. 138
Figure 6.3 Mean reading rate per sentence in milliseconds on the x-axis, vs. the percent of texts read at or under that rate on the y-axis, with one set of points/line per corpus section. The aggregate data from Figure 6.2 is depicted as a thick grey line. Sections that differ significantly from the overall data are plotted with squares representing data points. The x-axis is clipped at 5 seconds. Plot includes 95% of all the data.

Figure 6.4 Relative number of parsing states and reference mappings evaluated per sentence for each story when the relevant episode already exists in memory. Horizontal line at 1.0 indicates the value of no change.

Figure 6.5 Relative reference resolution and total time per sentence for each story when the relevant episode already exists in memory. Note the scales are not the same for both graphs. Horizontal line at 1.0 indicates the value of no change. The graph for reference resolution time also has a horizontal line at 2.0, indicating a double value in time.

Figure 6.6 Boxplots of the relative number of reference mappings evaluated per sentence for each story when the relevant episode already exists in memory. Horizontal line at 1.0 indicates the value of no change. Values below the line indicate a relative saving. Thick bars in the center of the boxplots represent median values. This graph includes only the 125 texts that changed.

Figure 6.7 shows the relative total time per text for each story when the relevant episode already exists in memory. Horizontal line at 1.0 indicates the value of no change. Values below the line indicate a saving in time when the episode exists in memory. This graph includes values for all 295 texts.

Figure 6.8 Histogram of differences in total time.

Figure 6.9 Histogram of differences in reference resolution time.
List of Tables

Table 2.1 Sizes of subsections of the corpus, including number of texts, average number of sentences, and average number of words per sentence. ................................................................. 42

Table 2.2 Sizes of subsections of the corpus, including number of texts, average number of sentences, and average number of words per sentence. Excluding texts used only to introduce new proper nouns, absent from the name recognition system. ................................................. 42

Table 3.1 Mappings to lexical and semantic concepts, with part of speech and agreement information, for the word “bombs.” .............................................................................................. 46

Table 5.1 Collections, predicates, and assertions in ResearchCyc accessible by REDMAP. .... 107

Table 5.2 Number of assertions produced for the core domains and the Lebanon corpora, and the correctness of those assertions. ............................................................................................................. 113

Table 5.3 Results of question answering evaluation for core domains corpus and the Lebanon corpus .......................................................................................................................................... 127

Table 6.1 Processing time and ambiguity for three texts with and without existing episodic knowledge. .................................................................................................................................. 147

Table 6.2 Amount of ambiguity when reading extended corpus with and without existing episodic knowledge for 125 texts that differed between conditions. ............................................. 153

Table 6.3 Processing time when reading the extended corpus with and without existing episodic knowledge. .................................................................................................................................. 156

Table 6.4 Processing time when reading the extended corpus with and without existing episodic knowledge for the 125 text subset depicted in table 6.2. ........................................................... 157
Chapter 1  Introduction

This dissertation presents research that focuses primarily on understanding descriptions of people, places, events, etc. described over multiple sentences and across multiple documents. The methods explored in this research produce understandings that are not only consistent with, but also integrated into a large existing knowledge base. The aim is to develop algorithms for understanding that can leverage and extend large existing structured memories, and to evaluate the role memory can play in language understanding.

Despite the availability of massive amounts of digital information, the creation of large structured knowledge bases, the Semantic Web, and many other technologies, there is still one key bottleneck for knowledge-based systems — computers can not, yet, sufficiently understand the vast majority of the world’s information. For the most part this is because they cannot comprehend language.

There is a whole spectrum of language understanding between seeing a bag of words and the deep understandings humans are capable of producing when they read. For example, it is one thing to know that a text contains the words “Barack Obama.” It is another to understand that a document written in June of 2009 using the phrase “the current President of the United States” is referring to the person we know to be named “Barack Obama.” It is yet another thing to understand the sentence, “Obama gave a speech at Notre Dame” as meaning that Barack Obama spoke at a university located in South Bend, IN. An even deeper understanding is needed to
recognizing that the previous example and “The commencement address at Notre Dame was delivered by the President” have similar meanings and could refer to the same event.

What constitutes sufficient understanding depends on the task, and language can be comprehended on many levels. For example, systems that retrieve documents given a set of query terms typically do not have representations of what they have read that are significantly deeper than basic word counts. These document retrieval systems have been able to deliver high degrees of success at their task with relatively shallow understandings of the documents they index. (Salton and Buckley 1988; Pinkerton 1994; Brin and Page 1998) In addition most of these systems do not have to identify all documents in their collection that are relevant to a user’s query, as long as they can find a document that is of use. They benefit from the fact that many ideas are written in many different ways throughout their collections. Powerset (Converse et al. 2008) is a new search engine that can form shallow semantic understandings at the sentence level and can recognize similar meanings in sentences, such as the last two examples about Obama speaking. It can also understand the difference between something Hillary Clinton said, and something said about Hillary Clinton, a distinction lost when only using word counts. This added knowledge and deeper understanding can improve performance and relevance to a query.

Other systems acquire facts from large collections of documents. These Information Extraction (IE) systems typically understand text at the phrase level, and extract relations such as capitals of countries, birthdays of famous people, etc. (Etzioni et al. 2005) These systems glean what they can from phrases they recognize but are able to ignore large portions of the text they process.
New research has gained traction on the problem by connecting the extraction process with a structured ontology so that relations between facts extracted can be leveraged. (Carlson et al. 2009) By adding this deeper level of understanding, these systems are able to improve their precision over systems without coupled semantic knowledge.

Document retrieval systems and information extraction systems have tasks that allow them to pick and choose what they attempt to understand, and therefore they can leverage shallow understanding techniques. Many language tasks would benefit from, if not require, a deeper level of understanding. Of primary interest to this dissertation are systems that need to understand the full contents of a text presented to them, such as learning by reading or intelligent tutoring. Full-text understanding is required to know what a particular author said about a particular topic, or to contrast two specific opinions. Other domains where there is limited writing or repetition that do not provide statistical methods with enough data are also relevant. The following section presents a typical example of the level of understanding targeted in this dissertation.

### 1.1 Example

A test corpus was created to test the ability of the system developed in this research to handle multi-sentence descriptions of people, places, events, etc. (A detailed description of the corpus used is provided in chapter 2.) For example, the following story describes some of the details of one of the events in the Israel-Lebanon conflict of 2006.
T1
Israel bombed the Beirut International Airport.
The attack occurred on July 13, 2006.
The bombing damaged 3 runways.
The Beirut International Airport was closed because of the bombing.

In order to understand this story, it is necessary to know that the bombing event implicit in the first sentence is the same as the event referred to by “the attack” in the second sentence and so on. It is necessary for a reader to connect these references to be able to correctly produce a representation of the text that contains only one bombing event, which occurred on the 13th and damaged 3 runways at the Beirut International Airport.

In addition to understanding descriptions that span multiple sentences, this research looks into how to extend and update understandings over time. For example, the Israel-Lebanon conflict spanned 34 days. Events pertaining to the conflict were constantly occurring and being reported on, updating the amount of information available to understand the situation. The event in story T1 occurred and was reported on immediately on July 13th, however it was not until the 14th in a subsequent story that the news reported why Israel targeted the airport. An excerpt from that reporting is presented as T2.

T2
Israel bombed the airport because Hezbollah used the airport for transporting weapons.

In order to understand that the airport referred to in story T2 is the Beirut International Airport it is necessary to understand that the events described in the two stories are in fact the same event.
Failing to make the connection between the events in the two stories could result in a reader building representations for more bombing events than actually occurred in reality. A fragmented representation would also prevent such a reader from explaining the reasons for the event in T1, or the details of the event in T2.

1.2 Understanding and Reference

One of the primary problems for computers to overcome when understanding language is ambiguity. Consider the following three example sentences.

I saw the man with the telescope.
The rabbit is ready for lunch.
Bush’s inauguration occurred in February.

Within the traditional models of research these three sentences present three different problems of ambiguity, typically handled in three different ways. The first presents a problem of phrase attachment (who has the telescope?), the second a problem of semantic interpretation (the rabbit is ready to eat, or to be eaten?), and the third illustrates episodic ambiguity (which Bush? and which inauguration event?). In addition to these ambiguities are the ambiguities presented in understanding examples T1 and T2, and in understanding that they refer to the same event. Integration with long-term memory further compounds the problem as ambiguities arise in connecting interpretations of the text with episodic knowledge and identifying a particular president, inauguration, or bombing in Lebanon.
Traditional models for understanding typically treat resolving each of these types of ambiguity with different methods. A claim of this dissertation is that each of these ambiguities as instances of a single problem, reference resolution, which is to identify to what concepts in memory the language is referring by resolving how references are grounded to existing semantic and episodic knowledge. The reference resolution algorithm I present handles all of these types of ambiguity simultaneously with a uniform mechanism, not only resolving references between semantic structures, but also grounding those references to existing episodic knowledge structures when and if they already exist in memory. Integration with the underlying memory is part of the reference resolution process, whereas it would be yet another step in the traditional model. A core component of this research is this reference resolution algorithm, and it is discussed in detail in chapter 4.

1.3 Pipeline Parsing

The most common approach to attempting deep language understanding is to break the problem up into subtasks that accomplish different levels of understanding. The processes that perform these subtasks are then stitched together to form a Natural Langue Processing (NLP) pipeline. (Russell and Norvig 1995; Allen 1995; Mitkov 2002) Each subtask brings different pieces of knowledge to bear, and typically uses different processing methods. For example, lexicons are leveraged early to perform stemming and word sense disambiguation, grammars are applied later to produce syntactic parse structures, which are in turn transformed into shallow semantic structures, and so forth.
The pipeline model for language understanding has the advantages of being modular and allowing researchers to focus on very specific and easily evaluated subtasks, however it succumbs to some significant limitations. One of the most severe is that some ambiguities at early stages of the pipeline can only be resolved with information from later stages, or in the worst case, only when the interpretation is merged with long-term memory at the very end of the pipeline. This means that either large amounts of ambiguity must be propagated through the pipeline, or choices must be made before enough information is actually available to properly decide.

In addition to the limitations of the practicality of this model with respect to ambiguity, are questions of how it approaches integration with memory. Schank (1982) and Riesbeck (1986) have long hypothesized the primacy of memory and expectation failure in understanding, and in this context, language understanding specifically. Only recently have neuroscience researchers through neuroimaging been able to experimentally investigate this claim in humans. Hagoort and van Berkum (2007) go beyond claiming that a one-stage model of language understanding is cognitively plausible, and further assert that given the evidence a pipeline model is not plausible.

1.4 Memory-Based Parsing

Memory-based parsing operates in stark contrast to the common pipeline model where syntactic structures are produced first and then used to construct or derive semantic interpretations. Integration with episodic memory is only performed in pipeline models after all other stages of understanding, and does not affect the processing of earlier pieces of the pipeline. Researchers
such as Schank (1999) claim not only that related knowledge and remindings must be retrieved from memory to perform langue understanding, but that performing the interaction with memory is the understanding. This interaction with memory is one of the core properties of a dynamic memory (Schank 1999). In much earlier research, Quillian, (1968) presented a model for language understanding called the Teachable Language Comprehender (TLC). In TLC semantic interpretation came first, and syntactic filters were only subsequently applied to confirm that a given semantic hypothesis was correct.

Direct Memory Access Parsing (DMAP) (Riesbeck and Martin 1986; Martin 1990) is a model for language understanding inspired by both the ideas of Quillian and Schank. DMAP understands language by using phrasal patterns to map text to its existing knowledge structures as early as possible during processing. DMAP is a model that primarily tries to recognize its input in terms of its existing knowledge, updating memory with new information only as needed. In fact the first implementation, DMAP-0 (Riesbeck and Martin 1986), could only recognize input, this task was seen as so central. DMAP was applied to multiple domains by both Martin (1990) and Fitzgerald (1994).

However, after this initial work, memory-based parsing has been relatively dormant for two decades (see section 7.3 for more on memory-based parsing). Recently there has been a resurgence in exploring memory-based approaches for language understanding, primarily in the context of the Learning by Reading community. This includes not only the work presented in
this dissertation, but also work by others such as (Etzioni 2007; McShane 2009; Nirenburg and Oates 2009). (See chapter 7 for a discussion of related work.)

1.5 Motivation

A large research initiative was started at Northwestern University to develop a learning-by-reading (LbR) system called the Learning Reader (LR) (Forbus et al. 2007). The goal of the research was to produce a system capable of reading text, integrating its interpretation of that text with long-term memory, and then thinking about what it has read off-line by generating analogies and asking itself questions. The system was also required to field parameterized questions or queries from users to discuss its knowledge and to evaluate the quality of its understanding.

The Learning Reader project had the goal of producing a machine that could acquire information from a variety of sources and integrate it with a large underlying knowledge base. It was to be capable of understanding descriptions that spanned multiple sentences and incorporating information over time from multiple sources. LR was to leverage and extend an existing large knowledge base. Knowledge accumulated by LR would be usable not just by itself but by any system capable of using the original knowledge base. This would allow the original knowledge base to be extended by acquiring knowledge from natural language. Given the tight coupling desired between the existing knowledge and new information acquired by LR, DMAP was a natural choice for implementing the required language understanding.
The research in this dissertation was used to implement the on-line text understanding portion of LR, the Reader. This system performs all of the types of understanding described earlier in this chapter. It can understand descriptions that span multiple sentences and integrate interpretations of evolving events over time with long-term memory. It does however have the following limitations. The Reader does not learn new vocabulary, except for proper names that are explicitly introduced. For example, a new person could be introduced with the text, “There is a person called ‘Hamid Karzai’.” This system also only learns about and extends specific instances. It does not build generalizations or manipulate its ontology based on what it has read. The off-line thinking portion of LR, the Ruminator, has explored building generalizations to a limited extent.

ResearchCyc, a subset of Cyc (Lenat 1995), was selected as the initial knowledge base for LR. ResearchCyc contains approximately 1.28 million assertions. This choice provided three unique challenges for DMAP that are the primary foci of this dissertation. First, the knowledge organized in ResearchCyc is logic-based, as opposed to the frame-based memories originally used by DMAP. Second, although it contains a lot of instance-based knowledge situated in a rich ontology, ResearchCyc is missing or does not connect its instances to high-level semantic packaging structures as the original memories used for DMAP did, for example, using scripts or memory organization packets (MOPs) (Schank 1999). Third, ResearchCyc is over three orders of magnitude larger than the frame-based knowledge bases originally used with DMAP systems, which raises interesting challenges of scale.
The original algorithms for DMAP were designed for systems operating on a frame-based memory. Multiple patterns referring to the same frame would facilitate understanding by specifying how references are resolved. Logic-based memories are structured differently, and when they are not populated with high-level semantic packaging structures, as is the case with ResearchCyc, new algorithms for resolving references are required, since there is not as much structured guidance for how sets of references are combined. All of these issues necessitated a new implementation of DMAP presented in this dissertation, called REDMAP (Reference resolution in Episodic memory for DMAP).

1.6 Reference Resolution in Episodic Memory

In this dissertation I presented an approach to language understanding that treats all ambiguity resolution as a problem of reference resolution: grounding references to episodic memory. This is different from traditional approaches to language understanding in two key ways. First, memory integration is an intrinsic part of this language understanding process beginning at the most early stages. Second, the DMAP approach makes language understanding fundamentally a problem of memory recognition.

In order to evaluate this idea I built an implementation of DMAP called REDMAP (Reference resolution in Episodic memory for DMAP) that resolves references dynamically and grounds them to existing knowledge structures as text is processed. The key ideas are, first that episodic memory can be used to facilitate connecting one reference in the text to another, and second that adding episodic memory is adding constraints that reduce ambiguity not multiply it.
REDMAP retrieves remindings to specific instances from episodic memory for the references in a text as soon as they are encountered. When reading a subsequent sentence and attempting to decide how to merge new references with the ones it has already seen REDMAP forms hypotheses about what would be true if the new references referred to the same concepts as existing references. REDMAP leverages the remindings to episodic memory it has already retrieved to evaluate these hypotheses. REDMAP accomplishes this by substituting remindings from earlier references in for the references in subsequent text. It then looks for the resulting knowledge structures in memory. If they are present in memory REDMAP concludes not only that the two references refer to the same thing, but also that they collectively refer to the specific episode in memory used in the query. In this way REDMAP treats reference resolution as a problem of reminding and lookup, not one of reasoning. Additionally, at the same time that REDMAP is identifying what references in the text corefer, it is also grounding those references to existing episodic memory.

1.7 Claims

The research presented in this dissertation is about performing reference resolution by grounding references in long-term episodic memory. Specifically it investigates ways an existing, large semantic and episodic memory can be leveraged to facilitate and even improve a reader’s understanding and performance. Performing this kind of understanding, grounding references to existing semantic and episodic knowledge structures on a large memory, raises serious questions regarding the scalability of this approach. A primary contribution of this dissertation is to
addresses these concerns and to evaluate the scalability of this approach to memory-based parsing.

Specifically in this dissertation I provide strong support for two claims.

**Claim 1:** A memory-based reference resolution algorithm (REDMAP) can provide broad coverage of and extending an existing large knowledge base by grounding to existing episodic memory as it parses and can use that memory to reduce ambiguity.

**Claim 2:** The reading rate (mean time per sentence for a text) of the REDMAP algorithm is empirically independent the number of references in the text and the length of input (number of sentences).

The evaluations provided in this dissertation also suggest an additional two claims which can be made with weaker support.

**Claim 3:** Complexity of syntax is not a predictor of processing time in REDMAP.

**Claim 4:** In domains that have fewer or less specific patterns REDMAP can piece together partial understandings at a cost to the accuracy and quantity of assertions produced. This relationship can be predicted to some degree from the amount of text understood.

After conducting this research some open questions still remain which are discussed later in this dissertation. This research was evaluated with a corpus of simplified English texts of which example T1 at the beginning of this chapter is representative (chapter 2 discusses this corpus and rational in detail). The evidence for claims 3 and 4 provides some support for ability of the
reference resolution algorithm to scale to more complicated texts, however it is still an open question to be evaluated. Unfortunately, in contrast to the clear evidence in support of claim 1, the savings obtained by REDMAP are currently mitigated by the cost of additional interaction with the knowledge base. However, REDMAP in the process of resolving references also performs knowledge integration, alleviating the need for this to be conducted as a separate step. Finally, the reference resolution algorithm currently only leverages existing episodic knowledge that is directly referred to by text being read. Future work would be to extend this algorithm to leverage generalizations, or similar episodic remindings when the knowledge that is being referred to does not already exist in memory.

1.8 Dissertation Overview

This dissertation is divided into eight chapters. Chapter 2 discusses the corpus of text used to evaluate REDMAP and LR and presents an introduction to the structure and organization of content in ResearchCyc.

Chapters 3 and 4 discuss how the REDMAP implementation of DMAP operates. Chapter 3 explains how REDMAP pattern matching rules map text to knowledge structures. It also discusses what knowledge is extracted from ResearchCyc to produce these pattern matching rules. Chapter 4 discusses how reference resolution is performed, including integration with the underlying knowledge base.
Chapters 5 and 6 provide evaluations of REDMAP. Chapter 5 focuses on the accuracy and quality of interpretations. Chapter 6 evaluates the scalability of REDMAP and provides the primary data for addressing the claims being made in this dissertation.

Chapter 7 presents related work, and chapter 8 provides concluding and summary remarks as well as discussion of future directions for this research.
Chapter 2 Knowledge Base and Corpus

As DMAP is a memory based parsing algorithm, it requires an underlying memory, or knowledge base on which to operate. Since a primary goal of this research was to evaluate DMAP’s ability to scale with respect to a large memory, a large knowledge base was needed. We selected ResearchCyc 1.0, a very large structured knowledge base containing approximately 1.28 million predicate logic assertions. This provides REDMAP with a large quantity of semantic and episodic knowledge to leverage, over two orders of magnitude more than what was available to previous frame-based DMAP systems.

To assess REDMAP’s ability to scale with a large memory it is important to have REDMAP retrieve and extend content in multiple areas of the knowledge base. A corpus of texts was selected which covered events, politics, and geography of the Middle East. The stories covered nine major topics: elections, geography, history, military actions, terrorist organizations, people (government and organization leaders), international relations, terrorist attacks, and the Israel-Lebanon conflict of 2006.

The remainder of this chapter introduces ResearchCyc, and the contents of the evaluation corpus.

2.1 Knowledge Base

ResearchCyc represents all of its knowledge using predicate logic assertions. Assertions are represented in a high-order logic called CycL, which is derived from first-order predicate
calculus (FOPC) and has some extensions specific to CycL. A more detailed description of CycL can be found in the Cyc 101 tutorial (Cycorp). The remainder of this section introduces the basic organization and terminology used in ResearchCyc that is relevant to REDMAP processing.

2.1.1 Collections

ResearchCyc represents objects and events in an extensive ontology with multiple layers of abstraction. For example, ResearchCyc represents that GovernmentLeader, Insurgent, and IraqiPerson are all subtypes of Person, and that Person is a subtype of both IntelligentAgent and BiologicalLivingObject. A depiction of a subset of the ResearchCyc ontology near Person is shown in figure 2.1. A type can have multiple subtypes and parent types. For example, the collection Person has 2,935 subtypes and 123 parent types. In ResearchCyc terminology a type is referred to as a collection, and the symbol representing a given collection in CycL represents the set of all instances of that type. For example, the symbol Person represents to the set of all people.

There are over 27,600 collections in the ResearchCyc ontology. In addition to representing objects and agents, the ontology also explicitly represents events. The generalization hierarchy for events is quite large with over 6,100 subtypes of the collection Event. A sample of few points in this section of the ontology is depicted in figure 2.2.
Figure 2.1 Subset of the ResearchCyc ontology near the collection Person. Arrows represent a generalization relationship between two collections.

Figure 2.2 Subset of the ResearchCyc ontology under the collection Event. Arrows represent a generalization relationship between two collections.

It should be noted that although symbols are given pseudo-English names, the symbol names themselves are opaque to the knowledge base. The names represent no information and are for the benefit of the humans interpreting their meaning; they are pneumonic for what the symbol is intended (but might not actually) represent. The symbols could all be changed to G01, G02, G03, etc. and represent the same content.
2.1.2 Relations

CycL assertions are represented as a set of symbols contained in parentheses written with the *predicate* first followed by a list of *arguments* that predicate requires. Predicates in CycL typically begin with a lowercase letter, while collections and instances start with capital letters. The generalization relation between collections in the ontology (depicted in figure 2.1 and figure 2.2) is expressed using the predicate *genls*. For example, the assertion *(genls Election Event)* represents that an *Election* is a subtype of *Event*.

A *genls* assertion connects each collection to its parent collections. For example, the following assertions connect the collection representing all events that are suicide bombings and also terrorist attacks to those three collections.

*(genls TerroristSuicideBombing Bombing)*
*(genls TerroristSuicideBombing SuicideAttack)*
*(genls TerroristSuicideBombing TerroristAttack)*

While the predicate *genls* is used to assert generalization relationships between collections, the predicate *isa* (“is a”) is used to represent a specific instance of a collection. For example, the following assertion states that Richard Nixon is an instance of the collection *GovernmentLeader*.

*(isa RichardNixon GovernmentLeader)*

The *isa* relation is transitive across the *genls* relation, so by extension of the above assertion it is also known that Nixon is a *Person*, an *IntelligentAgent*, and a *BiologicalLivingObject*. 
etc. (refer to figure 2.1). Instances can belong to multiple collections simultaneously, for example, to represent that Nixon is also male the following assertion would be made.

\[(\text{isa RichardNixon MaleHuman})\]

The ResearchCyc ontology also contains almost 9,000 predicates (sometimes referred to as \textit{relations}) for expressing relationships between concepts. For example, the following assertion represents that Nixon is a Quaker.

\[(\text{hasBeliefSystem RichardNixon QuakerReligion})\]

This example states the performer of an event.

\[(\text{performedBy September11TerrorAttackNewYork AlQaida})\]

Just as collections can represent generalizations of other collections, predicates also have a generalization hierarchy in ResearchCyc. For example, \texttt{performedBy} is a specific type of the predicate \texttt{doneBy}, the primary distinction being that \texttt{performedBy} indicates that the actor performed the action intentionally, while \texttt{doneBy} does not ascribe intentionality. An actor accidentally knocking over a glass would be represented with \texttt{doneBy}, while throwing a plate at someone would be more completely represented with \texttt{performedBy}. 
2.1.3 Non-Atomic Terms (NAT)

CycL also supports concept denoting functions, referred to as non-atomic terms (NAT). Frequently in our work, NATs are used to denote new collections, for example the function \texttt{GroupFn} is used to construct a collection that is the set of all sets of its argument. For example the following construct denotes the collection of all sets of soldiers.

\begin{verbatim}
(GroupFn Soldier)
\end{verbatim}

The following assertion refers to a specific group of soldiers.

\begin{verbatim}
(isa OneHundredFirstAirborn (GroupFn Soldier))
\end{verbatim}

NATs by themselves have no truth value, unlike assertions made with a predicate. NATs behave just like individual symbols, except their names are compound descriptors that have meaning. One value of a NATs such as \texttt{(GroupFn Soldier)}, is that it is then possible to simply reify a group of soldiers, without having the explicitly reify every single soldier (all of whom may be unknown) in the group. The value of this is even clearer, should one want to represent the set of all neurons that make up a brain. Another common use of NATs is to construct dates, for example the following, typically written in English as “June 28, 1979.”

\begin{verbatim}
(DayFn 28 (MonthFn June (YearFn 1979)))
\end{verbatim}

NATs may be used to represent anonymous, or previously unknown/unreified, instances, such as groups or dates, or NATs can be used as a functional way to refer to known entities. For
example, (ArmedForcesFn UnitedStatesOfAmerica) represents the instance that is the army, navy, etc. of the United States.

### 2.1.4 High-Order Relations

CycL also supports higher-order relations. Frequently second-order relations are used to encode general properties of collections. For example, the following assertion states that every instance of the collection Bombing, must have an instance of Bomb attached to it with the relation explosiveDeviceUsed.

\[(\text{relationAllExist explosiveDeviceUsed Bombing Bomb})\]

The next assertion states that it is a precondition for a bombing event that there is a gaining-control-of-a-bomb event.

\[(\text{preconditionFor-SitTypeSitType (GainingControlOfFn Bomb) Bombing})\]

ResearchCyc also contains a wealth of knowledge in the form of axioms like existentially quantified implications, such as if X is true then Y is true. REDMAP does not currently use this information.

### 2.1.5 Consistency and Microtheories

Knowledge bases, reasoning engines, and systems that use them, such as Learning Reader and REDMAP, typically require the knowledge they are operating on to be consistent.
Contradictions in the knowledge base are generally problematic; however inconsistencies frequently exist not only in how knowledge is represented, but also in the actual knowledge being represented. Inconsistencies range from conflicting theories on an aspect of physics, or simplifications that make assumptions, to differing reports on who performed an attack, or variations in analysis as to why the attack was performed.

ResearchCyc tackles the problem of consistency by partitioning the knowledge base into sets of assertions called microtheories. Microtheories should be self-consistent, but need not be consistent with each other. Multiple microtheories may also be used to represent various levels of detail for a given topic. For example, there can be a naïve-physics microtheory for representing basic physical properties of the world e.g., stuff falls when dropped, and a AP-physics microtheory for representing physics at the level of detail needed to pass a college course on the subject e.g., things accelerate at 9.8 meters per second when dropped on Earth.

Microtheories are formed into hierarchical knowledge contexts by connecting microtheories to parent microtheories. When reasoning or retrieving information from the knowledge base in the context of a microtheory, all the assertions in that microtheory are in scope as well as all the assertions in its parent microtheories. One important microtheory is BaseKB which contains the set of assertions core to ResearchCyc, including the upper layers of the ontology and general axioms. BaseKB has no parent microtheories. A second important microtheory is EverythingPSC (PSC stands for problem solving context), which inherits the assertions from every microtheory in the knowledge base. REDMAP uses this context when identifying
candidate knowledge references primarily for efficiency reasons. Due to REDMAP’s design this does not bring an increased risk of contradiction since it is primary querying to confirm the existence assertions already known to the knowledge base. See section 4.4.1 for additional discussion on the impacts of this decision.

2.2 Corpus Description

A goal of Learning Reader was to acquire a broad range of conceptual knowledge via reading while placing as few limitations as possible on the content of what is read. Although, there were no domain restrictions on the Learning Reader project as a whole, being able to generalize and compare stories, for example, via analogy, requires at least some relationship between the input stories. To evaluate scale with respect to a large knowledge base it was important to select input which covers a broad range of concepts present in the knowledge base, in this case, ResearchCyc. To include a broad range of content and satisfy the goals of the Learning Reader project, a corpus of texts were selected which covered events, politics, and the geography of the Middle East. An initial corpus was built from stories that covered eight major topics: elections, geography, history, military actions, terrorist organizations, people (government and organization leaders), international relations, and terrorist attacks. This corpus was generated at the beginning of the Learning Reader project, and is referred to as the core domains corpus. After the project was underway an additional ninth section was added covering the Israel-Lebanon conflict of 2006, referred to by itself as the Lebanon corpus. Collectively all nine sections, the core domains corpus plus the Lebanon corpus, are referred to as the extended corpus.
Stories for the Lebanon corpus were selected from a sequence of CNN articles spanning the conflict. Similarly election stories were produced from a set of CNN articles covering elections in Afghanistan, Iraq, and Pakistan. The terrorist organization stories were derived from a 2004 report located on the U.S. State Department’s Pattern of Global Terrorism website. (State department 2004) The other stories were selected from a set of articles from the website Information Please (Infoplease). The Infoplease website provided relatively straightforward coverage for a number of countries in the Middle East which is broken down into sections on geography, and history. We further separated the history section into pre-20th century (our history section of the corpus), and more modern events and politics, which became the other sections of our corpus. The people subsection of the corpus came from Infoplease articles on governmental and organizational leaders in the Middle East. Stories were selected that covered the history, geography, and politics of eight (8) Middle Eastern countries, eight (8) leaders of countries or organizations, three (3) terrorist organizations, elections in three (3) countries, ten (10) terrorist attacks, and six (6) military actions. The Lebanon corpus has content from thirty seven (37) different news articles.

2.2.1 Simplified English

Working from unconstrained text is an important challenge, however the goal of the Learning Reader project was to acquire, integrate, and reason about as much knowledge as could be extracted from a passage of text, from texts in a wide range of topics. The goal was to reach as much of the underlying knowledge base as possible, as opposed to covering as many syntactic or
surface forms of English as possible. To meet these goals we reduced the syntactic complexity of the input texts by rewriting them in simplified English. This was done so that research effort could be spent on scale and coverage of the knowledge base, and investigating how memory can be leveraged throughout the language understanding process. We were not interested in directly tackling the still very open question of resolving syntactic ambiguity. Section 6.3 provides additional discussion of how syntactic ambiguity affects REDMAP, and hints that we could have simplified the text less and probably have made the problem simpler for REDMAP. Further, as a line of research we also wanted to look at building a full end-to-end language understanding system to evaluate DMAP at scale. If REDMAP is viable in this context then future work can be done to expand throughput in terms of syntactic constructs.

The vocabulary was not restricted for constructing simplified English paraphrases, but a preference was made for simplified grammatical structure, as can be seen in the following example from the Lebanon corpus. The text is written using a large vocabulary but (generally) a relatively small number of simple English syntactic structures. The simplified stories were also intended to faithfully represent the content of the passages of the original story from which they were generated.

Corpus story:

Title: Israel authorizes 'severe' response to abductions
Date: Wednesday, July 12, 2006; Posted: 10:27 p.m. EDT (02:27 GMT)
Story:
JERUSALEM (CNN) -- The Israeli Cabinet authorized "severe and harsh" retaliation on Lebanon after Hezbollah guerillas kidnapped two soldiers and killed three others in a cross-border raid Wednesday.
Israel quickly blamed the Lebanese government for the raid -- and charged it with the soldiers' safe release -- and the Israel Defense Forces began hammering Lebanon with artillery and airstrikes hours before the Cabinet met to discuss a response.

…
Four Israel civilians and six soldiers have been wounded so far in the fighting, which has included more than 100 airstrikes on what Israel says are Hezbollah bases, and road and bridges that could be used in transporting the kidnapped soldiers.

Simplified English Paraphrase:

- Hezbollah attacked Israel.
- The attack killed 3 Israeli soldiers.
- Hezbollah kidnapped 2 Israeli soldiers.
- The attack occurred on July 12, 2006.
- The kidnapping occurred on July 12, 2006.
- Israel bombed Lebanon more than 100 times.
- The bombing was in response to the kidnapping.

The full CNN story had 954 words. From that story the three very dense sentences excerpted above were selected to be turned into the seven sentence simplified English paraphrase. Stories with multiple events typically become multiple texts, one for each event. Also not all information in a story was paraphrased, especially repeated content, which is common in a sequence of news stories regarding a developing event, as stories frequently start by recapping the events preceding the current story. Frequently stories contain content that would be challenging even for a skilled human to represent in ResearchCyc, and so that content is typically not included in the simplified English paraphrases. One example is the following sentence that appeared in the July 12, 2006 story excerpted above.
Earlier, Israel's chief of staff, Lt. Gen. Dan Halutz, told Israel's Channel 10, "If the soldiers are not returned, we will turn Lebanon's clock back 20 years."

2.2.2 Generating Simplified English

The core domains corpus stories were simplified by an undergraduate student associated with the project working part time for four weeks over the span of the Fall 2005 academic quarter. The student was instructed to, when possible, use active voice and minimize prepositional phrases while attempting to target the level of representation currently encoded in ResearchCyc. The student had access to the ResearchCyc lexicon, as well as a few in-house tools for searching for proper names as well as nouns and verbs with semantic interpretations in ResearchCyc. (More detail on this type of knowledge in ResearchCyc is presented in section 3.5.)

The 2006 Israel-Lebanon conflict section of the corpus was simplified in a similar manner by the author and another member of the Learning Reader project in July of 2007. Approximate 50 stories were collected from CNN written between July 12, 2006 and August 15, 2006. A subset of 37 stories was identified that covered the conflict and highlighted some of the causal relations being discussed. This was done in order to both push the level of what REDMAP could understand, and to provide input to other components of the Learning Reader interested in reasoning about causal relations. Examples of the kind of knowledge of interest in the Lebanon corpus include representing relationships between events, for example, that a particular bombing was performed in response to a kidnapping, as seen in the example in section 2.2.1, or
representing the reasons for a bridge being destroyed, i.e., to prevent the transportation of weapons across it.

After collecting and reading the CNN articles, it took approximately 30 person-hours to produce the simplifications. While this comes to a rate of about 9-man-minutes per sentence, it should be noted this process was far from direct. Two researchers were employed to generate a balanced corpus of specific content emphasizing causal relations, if the task were to simplify a stack of stories this process could likely be performed faster and would only require one person.

2.2.3 Breakdown of Corpus Contents

The selected stories were translated into a corpus of 423 simplified English texts. The sizes of the sections of the corpus divided by topic can be seen in table 2.1. See appendix A for an example text from each of the sections of the corpus.

Several sentences/texts in the corpus function only to introduce new named entities to REDMAP which were not already known to ResearchCyc. These texts are comprised primarily of sentences such as, “There is an airport called the ‘Beirut International Airport’.” Using separate texts to introduce new names was done to overcome engineering limitations in the lexical lookup operation and the way the pattern matcher recognizes new names in the same story. Future engineering effort could be applied to overcome this limitation. These are small texts and somewhat trivial from an understanding point of view. Table 2.1 includes those sentences and texts, while table 2.2 represents the corpus excluding them.
<table>
<thead>
<tr>
<th>corpus section</th>
<th>total texts</th>
<th>total sentences</th>
<th>total words</th>
<th>mean number of sentences per text</th>
<th>mean number of words per text</th>
<th>mean number of words per sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>election</td>
<td>26</td>
<td>69</td>
<td>468</td>
<td>2.7</td>
<td>18.0</td>
<td>6.8</td>
</tr>
<tr>
<td>geography</td>
<td>24</td>
<td>144</td>
<td>1162</td>
<td>6.0</td>
<td>48.4</td>
<td>8.1</td>
</tr>
<tr>
<td>history</td>
<td>129</td>
<td>292</td>
<td>2036</td>
<td>2.3</td>
<td>15.8</td>
<td>7.0</td>
</tr>
<tr>
<td>lebanon</td>
<td>87</td>
<td>209</td>
<td>1829</td>
<td>2.4</td>
<td>21.0</td>
<td>8.8</td>
</tr>
<tr>
<td>military</td>
<td>35</td>
<td>106</td>
<td>805</td>
<td>3.0</td>
<td>23.0</td>
<td>7.6</td>
</tr>
<tr>
<td>organization</td>
<td>27</td>
<td>69</td>
<td>438</td>
<td>2.6</td>
<td>16.2</td>
<td>6.5</td>
</tr>
<tr>
<td>person</td>
<td>41</td>
<td>110</td>
<td>732</td>
<td>2.7</td>
<td>17.9</td>
<td>6.7</td>
</tr>
<tr>
<td>relation</td>
<td>34</td>
<td>106</td>
<td>916</td>
<td>3.1</td>
<td>26.9</td>
<td>8.6</td>
</tr>
<tr>
<td>terrorism</td>
<td>20</td>
<td>82</td>
<td>555</td>
<td>4.1</td>
<td>27.8</td>
<td>6.8</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td><strong>423</strong></td>
<td><strong>1187</strong></td>
<td><strong>8941</strong></td>
<td><strong>2.8</strong></td>
<td><strong>21.1</strong></td>
<td><strong>7.5</strong></td>
</tr>
</tbody>
</table>

Table 2.1 Sizes of subsections of the corpus, including number of texts, average number of sentences, and average number of words per sentence.

<table>
<thead>
<tr>
<th>corpus section</th>
<th>total texts</th>
<th>total sentences</th>
<th>total words</th>
<th>mean number of sentences per text</th>
<th>mean number of words per text</th>
<th>mean number of words per sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>election</td>
<td>16</td>
<td>59</td>
<td>403</td>
<td>3.7</td>
<td>25.2</td>
<td>6.8</td>
</tr>
<tr>
<td>geography</td>
<td>8</td>
<td>126</td>
<td>1045</td>
<td>15.8</td>
<td>130.6</td>
<td>8.3</td>
</tr>
<tr>
<td>history</td>
<td>86</td>
<td>250</td>
<td>1753</td>
<td>2.9</td>
<td>20.4</td>
<td>7.0</td>
</tr>
<tr>
<td>lebanon</td>
<td>86</td>
<td>208</td>
<td>1823</td>
<td>2.4</td>
<td>21.2</td>
<td>8.8</td>
</tr>
<tr>
<td>military</td>
<td>20</td>
<td>92</td>
<td>713</td>
<td>4.6</td>
<td>35.7</td>
<td>7.8</td>
</tr>
<tr>
<td>organization</td>
<td>11</td>
<td>53</td>
<td>335</td>
<td>4.8</td>
<td>30.5</td>
<td>6.5</td>
</tr>
<tr>
<td>person</td>
<td>20</td>
<td>89</td>
<td>604</td>
<td>4.5</td>
<td>30.2</td>
<td>6.8</td>
</tr>
<tr>
<td>relation</td>
<td>30</td>
<td>102</td>
<td>886</td>
<td>3.4</td>
<td>29.5</td>
<td>8.7</td>
</tr>
<tr>
<td>terrorism</td>
<td>18</td>
<td>80</td>
<td>542</td>
<td>4.4</td>
<td>30.1</td>
<td>6.8</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td><strong>295</strong></td>
<td><strong>1059</strong></td>
<td><strong>8104</strong></td>
<td><strong>3.6</strong></td>
<td><strong>27.5</strong></td>
<td><strong>7.7</strong></td>
</tr>
</tbody>
</table>

Table 2.2 Sizes of subsections of the corpus, including number of texts, average number of sentences, and average number of words per sentence. Excluding texts used only to introduce new proper nouns, absent from the name recognition system.
Since 128 of the 423 total texts functioned only to add new names and were structured in essentially the same way as each other, including them would skew the results towards texts of this type and structure. Unless otherwise noted data presented throughout this dissertation is generated from the subset of 295 stories depicted in table 2.2.
Chapter 3  Mapping Text to Semantic and Episodic Structures

One of the core processes of REDMAP is mapping text to semantic and episodic structures. REDMAP has two main ways of connecting text to knowledge structures. The first is by using a lexicon to map words to lexical and semantic structures in the knowledge base, as well as link proper nouns to existing episodic instances. The second is by using pattern matching rules to connect phrasal patterns to structures in memory. In cases where text is ambiguous REDMAP will entertain multiple hypotheses about its meaning, allowing later text or episodic memory to disambiguate. The following chapter discusses in detail how REDMAP maintains multiple hypotheses and resolves ambiguity. This chapter is about how text is mapped to meaning.

Producing a lexicon and a set of rules large enough to cover a significant portion of the knowledge base would be very labor intensive. Cycorp has already generated a large body of linguistic knowledge which is included in ResearchCyc. This knowledge was extracted to produce a lexicon and a set of pattern matching rules that REDMAP needs to operate. The remainder of this chapter discusses how the mappings in the REDMAP lexicon and rules are used, and what knowledge was extracted from ResearchCyc in order to bootstrap REDMAP.

3.1 Lexicon

Input into REDMAP is tokenized into words. Entries in the lexicon are then identified that map those words to existing semantic and episodic content. This is done in two ways. The first is to recognize a known list of proper names. The second is for all other words.
Mappings in the lexicon connect proper names to a specific named concept in episodic memory. For example, the lexicon maps text like “George W. Bush” to the known episodic instance \texttt{GeorgeWBush}. Some references are ambiguous, for example “George Bush” has two mappings in the lexicon: one to \texttt{GeorgeWBush} and one to \texttt{GeorgeHWBush}.

Other text, such as common nouns and verbs cannot be mapped as directly, but the process is still relatively straightforward. Since REDMAP is built on top of ResearchCyc we have adopted some of the linguistic structures created by Cycorp. ResearchCyc has a set of \textit{lexical concepts} which can be used to represent words. For example, the word “bomb” is represented in ResearchCyc with the lexical concept \texttt{Bomb-TheWord}. Lexical concepts represent morphological roots, and so multiple stems will map to the same concept. For example “bomb”, “bombed”, and “bombs” all map to \texttt{Bomb-TheWord}. In this example, REDMAP also records that the first mapped to the lexical concept directly, the second followed a stemming path by removing the “-ed” suffix, and similarly the third with the “-s” suffix. ResearchCyc also contains mappings from lexical concepts to semantic concepts. Based on the stemming path used REDMAP can identify the correct semantic denotation, disambiguating between word senses, for example between the ResearchCyc concepts \texttt{Bomb}, \texttt{Bombing}, and \texttt{Bomber}.

Given the input “bombs” REDMAP will identify the following lexical concepts and denoted semantic concepts annotated with part of speech and agreement information.
<table>
<thead>
<tr>
<th>Concept</th>
<th>Part of Speech</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bomb-TheWord</td>
<td>noun</td>
<td>3rd-plural</td>
</tr>
<tr>
<td>(GroupFn Bomb)</td>
<td>noun</td>
<td>3rd-plural</td>
</tr>
<tr>
<td>Bomb-TheWord</td>
<td>verb</td>
<td>3rd-singular</td>
</tr>
<tr>
<td>Bombing</td>
<td>verb</td>
<td>3rd-singular</td>
</tr>
</tbody>
</table>

Table 3.1 Mappings to lexical and semantic concepts, with part of speech and agreement information, for the word “bombs.”

Agreement information in REDMAP is represented in a manner similar to Norvig (Norvig 1992).

### 3.2 Rules

The second way REDMAP maps text to knowledge structures is with rules. Rules have two primary components, a pattern, and the semantic or episodic structure the pattern maps to, which is referred to as the rule’s base. When a rule pattern is matched REDMAP generates a reference to the semantic and episodic content in its base. Rules are used to map a sentence like “Israel bombed Lebanon” through a pattern like “<agent> bomb <target>” into the following set of assertions.

```prolog
((isa ?action Bombing)
 (performedBy ?action Israel)
 (objectHarmed ?action Lebanon))
```

The symbol ?action represents some specific bombing that has yet to be grounded in the knowledge base.
3.2.1 Rule Phrasal Patterns

Rule patterns can match sequences of text and lexical, semantic, and episodic concepts produced from mappings in the lexicon, or in other rules. The following is an example of a simple pattern for recognizing the concept military airplane, which matches two pattern parts, a text string followed by a lexical concept.

```
(rule
 :name rule-military-Airplane
 :pattern ("military"
   (:and Airplane-TheWord
     (:pos :noun)
     (:agree ?agreement)))
 :base ((isa ?x MilitaryAirplane))
 :bindings ((?agreement . :3rd-singular)))
```

This pattern expects the string “military” followed by the lexical concept `Airplane-TheWord`, which REDMAP will produce from mappings in the lexicon when it sees the word “airplane.” This pattern is matched with the following two parts.

```
"military"
Airplane-TheWord :pos :noun :agreement :3rd-singular
```

In addition to matching concepts, rule patterns may also specify constraints on the properties of the concepts that can match. For example, patterns can specify part of speech or agreement requirements. The above pattern requires the reference to the lexical concept `Airplane-TheWord` to be a noun and have third person singular agreement. Thus this rule will only matches the phrase “military airplane”, not “military airplanes”, as the later is plural. (For a discussion of how this rule was extracted from ResearchCyc see section 3.3.2.2.)
When this rule is matched REDMAP will produce a reference to an instance of the collection MilitaryAirplane. The specific instance is yet to be identified and is represented with a variable that is not yet grounded in episodic memory. The process for fully resolving references in memory is discussed in the next chapter.

The previous examples illustrate patterns which can match strings and lexical concepts, the next example is a pattern which requires matching semantic concepts. (The following example, continued in section 3.2.2, discusses the rule rule-s-Bomb-o that was extracted from ResearchCyc, more information on this can be found in section 3.3.2.1.) The following pattern represents using “bomb” as a transitive verb, e.g., <subject> bomb <object>, for example “Israel bombed Lebanon,” or “Terrorists bombed a building.” Using “bomb” as a transitive verb implies and requires that the subject be some kind of agent performing the bombing, and that the object of the verb be some kind of object that is being blown-up. In ResearchCyc the broadest collection that represents agents is Agent-Generic, and the broadest collection containing objects which could be blown-up is SomethingExisting. A REDMAP rule pattern for this case would expect three parts: a reference to an instance of the semantic concept Agent-Generic, the lexical concept Bomb-TheWord, and a reference to an instance of the collection SomethingExisting. The following pattern meets these requirements.
(rule
  :name rule-s-Bomb-o
  :pattern ((:and (isa ?subject Agent-Generic)
                (:pos :noun)
                (:agree ?agreement))
           (:and Bomb-TheWord
                (:pos :verb)
                (:agree ?agreement))
           (:and (isa ?object SomethingExisting)
                (:pos :noun)))
  :base <see section 3.2.2>)

In addition to matching the three concepts, this pattern specifies that the concepts matching the subject and object be nouns, and that the lexical concept Bomb-TheWord must be used as a verb. Further it requires that the subject and the verb agree. When the subject is matched its value for agreement will be bound into the variable ?agreement, which initially has no value, then when the verb is seen its agreement must also unify with the value of the variable ?agreement which now contains the agreement from the subject.

The following discussion traces how the two example sentences are matched by the rule pattern just described.

**T1** Israel bombed Lebanon.

**T2** Terrorists bombed a building.

Mappings (also referred to as references), to semantic and episodic concepts in REDMAP are organized in a data structure called a *marker*. This name is largely historical, and comes from the connection to the marker passing algorithms used in the original DMAP implementations. The primary field of a marker is its *base* which indicates the linguistic, semantic, and episodic
content to which the marker is mapping. The base is sometimes referred to as a *base set* when there are multiple assertions; bases represent conjunctions of all their parts. A marker’s base can contain variables, represented as terms that begin with a question mark, and those variables are bound to values or have additional constraints associated with them depending on how the rules that produced those markers were matched. The following discussion provides some examples of this. Markers also contain additional properties, for example part of speech and agreement annotations.

For the sentence T1, REDMAP will produce the following markers from mappings in the lexicon.

M1 (marker
   :base (Israel)
   :pos :noun
   :agreement :3rd-singular)

M2 (marker
   :base (Bomb-TheWord)
   :pos :verb
   :agreement :finite-past))

M3 (marker
   :base (Lebanon)
   :pos :noun
   :agreement :3rd-singular)

The first marker, M1, is a reference to the known instance in memory for the concept *Israel*, and was generated though a mapping in the lexicon between that concept and the string “Israel.” Since this mapping was made through the use of a proper noun, the marker mapping to the concept *Israel* also inherits properties of the text used to generate the mapping, namely that it
was a noun and is a third-person, singular reference. The markers M2 and M3 are generated similarly.

Looking at the collections that Israel is a member of in the knowledge base, REDMAP knows the assertion

\[(\text{isa} \text{ Israel Agent-Generic})\]

to be true and therefore is able to match M1 with the first part of the pattern, repeated below.

\[(:\text{and} \ (\text{isa} \ ?\text{subject Agent-Generic})
\quad (\text{pos} \ :\text{noun})
\quad (:\text{agree} \ ?\text{agreement}))\]

In matching this part of the pattern the variable \(?\text{subject}\) will be bound to the value \text{Israel}. M1 matches the other two constraints on the first part of the pattern as it is annotated as a noun, and the agreement property of M1 unifies with (and is bound to) the rule’s \(?\text{agreement}\) variable. Similarly M2 matches the second part of the pattern, and the agreement value for M2 unifies with the agreement value of M1, now stored in the value of \(?\text{agreement}\). M3 matches the third part of the pattern in a manner similar to how M1 matched the first part, and in doing so \(?\text{object}\) is bound to the value \text{Lebanon}. Figure 3.1 depicts how concepts generated from the text are used to match this pattern. The edges are labeled with knowledge in ResearchCyc which allowed the connections to be made.
The previous example shows how the example rule can match specific episodic instances (e.g., Israel and Lebanon). The following example illustrates how the same rule can match text which refers to semantic references generated by common nouns (e.g., terrorists and building).

Mappings in the lexicon for the words in the sentence T2 allow REDMAP to produce the following markers.
Figure 3.2 Diagram of T2 matching rule-s-Bomb-o, primary components bolded. Links are annotated with assertions from ResearchCyc that facilitated the connection.

M4 (marker
  :base ((isa ?x TerroristGroup))
  :pos :noun
  :agreement :3rd-plural)

M5 (marker
  :base (Bomb-TheWord)
  :pos :verb
  :agreement (:finite ? :past))

M6 (marker
  :base ((isa ?x Building))
  :pos :noun
  :agreement :3rd-singular)
The marker M4 will match the first part of the pattern in the rule *rule-s-Bomb-o*, which expects a reference to an Agent-Generic. The marker contains a reference to a TerroristGroup, and the assertion

\[(\text{gens} \text{TerroristGroup Agent-Generic})\]

is true in ResearchCyc, stating that the collection TerroristGroup is a subtype of Agent-Generic, thus the pattern is satisfied. Again M5 matches as did M2 in the previous example. Finally M6 will match the third part of the pattern in the same way M4 matched the first part of the pattern. Figure 3.2 depicts this pattern being matched.

### 3.2.2 Rule Bases

After a rule pattern is matched the rule base will be instantiated as a new marker. Since rule patterns can match a variety of inputs, variables in the pattern bind to the values they match, allowing this content to be referred to in the base. The rule *rule-s-Bomb-o* is depicted below including its base. The base refers to three variables, two of which, ?subject and ?object, will have values bound to them when the pattern is matched. The third variable, ?action, represents the bombing event itself and is a new, open reference which will be in the marker produced by this rule.
(rule-s-Bomb-o
  (pattern ((and (isa ?subject Agent-Generic) (:pos :noun) (:agree ?agreement)) (and Bomb-TheWord (:pos :verb) (:agree ?agreement)) (and (isa ?object SomethingExisting) (:pos :noun))))
)

When this rule is matched by the sentence “Israel bombed Lebanon” (as with the example T1 using the markers M1, MC10, and M3) a new marker will be produced that looks like the following.

M7 (marker
  :bindings ((?subject . Israel) (?object . Lebanon)))

The specific instances matched in the rule become part of the marker, which is equivalent to producing a marker with the following base.

((isa ?action Bombing) (performedBy ?action Israel) (objectHarmed ?action Lebanon))

This base refers to some specific, but yet to be grounded in the knowledge base, instance of a bombing performed Israel that harmed Lebanon.
A slightly different marker will be produced from matching the second example, T2 (matching markers M4, MC13, and M6). In this case specific instances were not provided by the text, but the matched markers still contain additional information that constrains the mapping being produced that must be included. Since the variable \(?subject\) was matched by the variable \(?x\) in marker M4, an instance of the collection \(\text{TerroristGroup}\), then \(?subject\) must also be constrained to be an instance of the collection \(\text{TerroristGroup}\), not just a member of the far more general collection \(\text{Agent-Generic}\). Similarly from the information in M6, \(?object\) will be constrained to be an instance of the collection \(\text{Building}\). This results in the creation of the following marker.

\[
\text{M8 (marker)} \\
\hspace{1cm} :\text{base} \ (\text{isa} \ ?\text{action} \ \text{Bombing}) \\
\hspace{2cm} (\text{performedBy} \ ?\text{action} \ ?\text{subject}) \\
\hspace{2cm} (\text{objectHarmed} \ ?\text{action} \ ?\text{object}) \\
\hspace{2cm} (\text{isa} \ ?\text{subject} \ \text{TerroristGroup}) \\
\hspace{2cm} (\text{isa} \ ?\text{object} \ \text{Building}))
\]

The following two original isa assertions are removed, as they are redundant to the new information provided by the markers that matched the pattern.

\[
(\text{isa} \ ?\text{subject} \ \text{Agent-Generic}) \\
(\text{isa} \ ?\text{object} \ \text{SomethingExisting}))
\]

M8 maps to some specific, but yet to be grounded in the knowledge base, instance of \(\text{Bombing}\) that was performed by a specific, but yet to be grounded, instance of \(\text{TerroristGroup}\), and that targeted a specific, but again, yet to be grounded, instance of \(\text{Building}\).
3.2.3 Nested Pattern Matching Example

The example sentence T3 is made of up of two phrases. While it is possible to create an "<agent> target <target> in <location>" pattern and rule, this approach would quickly become expensive, as every verb which could take "in <location>" as a prepositional phase would have to be replicated for this construction. Generating rules for every possible sentence configuration is intractable, if not impossible.

T3 Al-Qaida targeted US soldiers in Afghanistan.

Instead compositional rules are needed that could match phrases, such as "<event> in <location>." REDMAP rules operate this way and can match hierarchically, allowing references represented in a marker produced after one rule matches to be used to satisfy part of the pattern being expected by another rule. The remainder of this section presents a hierarchical rule matching example.

The sentence T4 is the same as the first half of sentence T3. T4 is matched by the rule rule-s-Target-o (depicted below) that represents the semantics for the sense of the verb “target” where an agent is (intending to) attack an object. (This rule was produced manually, and was modeled from an existing rule for Attack-TheWord that produced the same semantics. ResearchCyc did not contain this sense of the verb “target.”)

T4 Al-Qaida targeted US soldiers.
Although the rule pattern and the sentence do not explicitly refer to an attack, there is an implicit attack event being discussed. This event is referred to in the base of the rule by the variable ?action.

A marker produced by this rule will contain a reference to an attack event in its base. This event could then be used to satisfy a pattern that expects a reference to an event. The following rule, rule-Event-In-Loc, matches the phrase "<event> in <location>," and was extracted from knowledge in ResearchCyc.

Expanding the example from T4 back to the original sentence T3 now creates a situation where the two rules described will interact with each other. The first four words of T4 will produce
markers which will satisfy the pattern for rule-s-Target-o which then results in the production of the following marker.

**M9** (marker
- base ((isa ?action AttackOnObject)
  (performedBy ?action ?subject)
  (intendedAttackTargets ?action ?object)
  (isa ?subject Agent-Generic)
  (isa ?object
    (GroupFn
      (MemberFn
        (ArmyFn UnitedStatesOfAmerica)))))
- bindings ((?subject . AlQaida)))

Marker M9 matches the first part of the pattern for rule-Event-In-Loc because it refers to an instance of AttackOnObject which is a type of Event. The variable ?action in M9 will unify with the variable ?event in the rule pattern. The rest of the rule pattern will then match the last two words in the example sentence T3. This entire example is depicted in Figure 3.3.

The depictions of markers produced by rules up until now have shown constraints from the pattern matching being merged into the marker being produced by the rule. This is a slight oversimplification. As the system is currently implemented, each marker actually has a table to record dependencies to other markers for each of its variables. In the above example, when rule-Event-In-Loc is matched and produces a marker, that marker will contain a table indicating that its ?event variable is connected to, and is dependent on, the ?action variable from M9 since that is what was used to match the pattern. Thus REDMAP records that those two variables corefer. More discussion of how references are tracked is presented in the next chapter.
3.3 Knowledge Extracted from ResearchCyc

The ResearchCyc ontology has numerous predicates for mapping natural language constructs to ground instances, and collections, as well as sets of predicates which provide information for the interpretation of verb phrases or sentences in ResearchCyc semantics. This information was extracted and incorporated into REDMAP. The assertions extracted from ResearchCyc are either transformed into REDMAP pattern matching rules, or directly incorporated into the REDMAP lexicon and proper name recognizer. The linguistic knowledge extracted has provided REDMAP with over 64,000 proper names, over 36,000 words, and over 180,000 rules.
The remainder of this chapter discusses the extraction of knowledge from ResearchCyc in greater detail.

### 3.3.1 Extracted Lexicon Content

All of the following content is incorporated directly into the REDMAP lexicon, which is described in section 3.1.

#### 3.3.1.1 Proper Names

Names are stored in ResearchCyc with the predicate `nameString`, or its specializations. The following represents a few examples.

```
(nameString GeorgeWBush "George Bush")
(nameString Afghanistan "Afghanistan")
(nameString MadridTrainBombings2004 "Madrid train bombings")
```

This added 64,000 noun phrases referring to 43,000 distinct entities ResearchCyc to REDMAP’s lexicon.

#### 3.3.1.2 Lexical Concepts

ResearchCyc represents words as lexical concepts, and there are assertions which map word strings to these concepts, for example, “bomb” is represented as the ResearchCyc lexical concept `Bomb-TheWord`. Irregular conjugations are also recorded in the knowledge base, however, it is
up to the natural language system using ResearchCyc information to know that “bombs” is the plural of Bomb-TheWord using the standard rules of English (stemming). Some examples of these assertions are the following.

(infinitive Bomb-TheWord “bomb”)
(singular Bomb-TheWord “bomb”)

(singular Person-TheWord “person”)
(plural Person-TheWord “people”)

(infinitive Fly-TheWord “fly”)
(pastTense-Universal Fly-TheWord “flew”)
(perfect Fly-TheWord “flown”)
(agentive-Sg Fly-TheWord “flyer”)
(agentive-Sg Fly-TheWord “flier”)
(singular Fly-TheWord “fly”)

This adds 36,000 English words referring to 19,000 distinct lexical concepts to the REDMAP lexicon.

3.3.1.3 Denotations to Semantic Concepts

Additional assertions in ResearchCyc map lexical concepts to semantic concepts. For example, the string “attack” maps to the lexical concept Attack-TheWord, another assertion indicates that when Attack-TheWord is used as a (count) noun it can be mapped to the ResearchCyc semantic concept AttackOnObject (a type of Event in the ontology). The following are examples of denotation assertions provided by ResearchCyc. The predicate denotation maps a lexical concept used as a particular part of speech to a semantic concept. The third argument is a number to assist in tracking which word-sense is being represented, it is irrelevant for REDMAP and can be ignored for this discussion.
Extracting this information gives REDMAP the ability to map over 8,000 lexical concepts to over 10,800 semantic concepts. One lexical concept can map to multiple semantic concepts and vice versa.

3.3.2 Extracted Rules

All other linguistic information extracted from ResearchCyc is used to produce REDMAP pattern matching rules. This includes information about adjectives, multi-word nouns and noun phrases, and especially verb phrases. A knowledge extraction language was created to write templates to translate from the type of knowledge stored in ResearchCyc to a set of rules structured for REDMAP. Each rule extractor written could thus apply to numerous assertions in the ResearchCyc. Extractors frequently produced thousands of rules, and some extractor templates generated over 30,000 rules each. In total 64 extractors were written to mine over 180,000 rules from ResearchCyc.
3.3.2.1 Verbs

REDMAP gets the bulk of its deep semantic leverage from the ResearchCyc linguistic information related to verbs. These assertions in ResearchCyc map uses of a verb to templates for a set of CycL assertions that has place holders as needed for the subject, object, event(/action), direct object, etc. An example of such an assertion follows.

```
(verbSemTrans Bomb-TheWord 0 TransitiveNPFrame
 (and
  (isa :ACTION Bombing)
  (performedBy :ACTION :SUBJECT)
  (objectHarmed :ACTION :OBJECT)))
```

This assertions states that when “bomb” is used as a transitive verb, *i.e.*, <subject> Bomb-TheWord <object>, for example “Israel bombed Lebanon,” it can be interpreted using the following semantic template.

```
(isa :ACTION Bombing)
(performedBy :ACTION :SUBJECT)
(objectHarmed :ACTION :OBJECT)
```

REDMAP patterns can match ResearchCyc lexical concepts like Bomb-TheWord, however, the other references like :SUBJECT and :OBJECT need to be transformed into expectations for the REDMAP patterns. These keyword references (:SUBJECT, :OBJECT, etc.) are designed for the way syntactic parsers, like the Link parser (Sleator and Temperley 1993) and Charniak’s parser (2000), identify and manipulate parsing structures. The keywords provide mappings between the text and the semantics. Where they can occur in the text is indicated by the linguistic frame type,
for example TransitiveNPFrame. This example specifies that a transitive verb will be used with a subject and an object that is a noun-phrase.

To identify appropriate expectations for these references in REDMAP the rule extraction algorithm looks at how they are used in the semantic template. In this example :SUBJECT is used as the second argument in the performedBy predicate allowing the extractor to derive its type from other ResearchCyc assertions about the structure of the ontology and its semantics. For example the following assertion in ResearchCyc states that the second argument of a performedBy relation must be of type Agent-Generic.

\[(\text{argIsa performedBy 2 Agent-Generic})\]

Similarly for :OBJECT assertions specify the second argument of objectHarmed is of the type SomethingExisting. This allows REDMAP to set expectations for references to specific semantic type instead of accepting any structure that merely fits the syntactic constraints, for example <Agent-Generic> Bomb-TheWord <SomethingExisting>, instead of <subject> Bomb-TheWord <object>. During rule extraction if there are multiple references to a variable in the semantic template only the most specific ones will be used. If there are multiple references which are not on a common branch in the ontology, all referenced leaf types will be required.

The REDMAP rule created from the verbSemTrans assertion for Bomb-TheWord, shown above, is the following.
This rule has a pattern expecting three references: a reference to an Agent-Generic, the lexical concept Bomb-TheWord, and finally a reference to a SomethingExisting. Further the pattern expects that the reference to Agent-Generic be annotated as a noun, and that Bomb-TheWord be annotated as a verb, and that they both agree with each other in terms of tense and count. All of this information is placed in the rule by the rule extraction algorithm using the template it was given.

Verb-related assertions in ResearchCyc were mined to produce 5,413 REDMAP rules.

### 3.3.2.2 Multi-word Nouns

In addition to the noun information incorporated in the lexicon, there is additional linguistic information in ResearchCyc for multi-word noun phrases, which is mined to produce REDMAP rules. An example of a noun rule would be the conversion of the following ResearchCyc assertion into a set of pattern matching rules.
It is important to distinguish between singular or plural references to nouns. One alternative is to engineer the pattern matcher to be sensitive to this information, the other is to simply replicate the rule for each case. We took the simpler approach and replicated the rules. The following two rules produce interpretations for “military airplane” and “military airplanes” respectively.

The noun rules discussed up until this point have all produced a reference to an instance of a collection as a base using a single isa assertion. However there is no restriction on the content which can be included in a rule’s base. The following assertion in ResearchCyc represents the semantics for a “terrorist training camp”.

```
(multiWordString (TheList "military") Airplane-TheWord
CountNoun
MilitaryAirplane)
```

```scheme
(rule
:name rule-military-Airplane
:pattern ("military"
 (:and Airplane-TheWord
  (:pos :noun)
  (:agree ?agreement)))
:base ((isa ?x MilitaryAirplane))
:bindings ((?agreement . :3rd-singular)))

(rule
:name rule-military-Airplane-2
:pattern ("military"
 (:and Airplane-TheWord
  (:pos :noun)
  (:agree ?agreement)))
:base ((isa ?x (GroupFn MilitaryAirplane)))
:bindings ((?agreement . :3rd-plural)))
```
The rule extractor for \textit{CountNoun} instances of \textit{multiWordSemTrans} produced the following REDMAP rule.

\begin{verbatim}
(rule
 :name rule-terrorist-training-Camp
 :pattern ("terrorist" "training"
   (:and Camp-TheWord
     (:pos :noun)
     (:agree ?agreement))
 :base
   ((relationExistsInstance eventOccursAt
     (TrainingForFn TerroristAct) :NOUN))
   (isa ?x TrainingCamp)
 :bindings ((?agreement . :3rd-singular)))
\end{verbatim}

This rule produces a reference to an instance that is a member of the collection \textit{TrainingCamp}, but further constrains that instance using a higher order relation to represent that training for terrorist activities occurs at that training camp instance.

Noun-related lexical assertions in ResearchCyc were mined to produce 165,914 REDMAP rules.
3.3.2.3 Adjectives

Several types of assertions in ResearchCyc provide information about how to represent adjectives. Some examples of these assertions and the rules they produce are the following.

(adjSemTrans
 Arid-TheWord
 1
 RegularAdjFrame
 (ambientRelativeHumidity :NOUN
  (VeryLowAmountFn RelativeHumidity)))

The extractor processing this assertion produced the following rule, which could match phrases such as “arid desert” or “arid country”.

(rule
 :name rule-Arid-n
 :pattern (Arid-TheWord
  (:and (isa ?noun PartiallyTangible)
  (pos :noun)))
 :base ((ambientRelativeHumidity ?noun
  (VeryLowAmountFn
  RelativeHumidity))
  (isa ?noun PartiallyTangible))

Another common construct, especially in geography stories, is to discuss where geographic features or countries are located relative to each other. The following example is for understanding things like “The arid desert land is west of the Euphrates.”

(adjSemTrans
 West-TheWord
 0
 (PPCompFrameFn TransitivePPFrameType Of-TheWord)
 (permanentlyEastOf :OBLIQUE-OBJECT :NOUN))
The ResearchCyc assertion indicates that this is representing the use of “west” where the oblique object is attached with an “of” prepositional phrase. The extractor processing this assertion produced the following rule.

```plaintext
(rule
 :name rule-n-West-Of-o
 :pattern ((isa ?noun GeographicalThing)
           West-TheWord
           Of-TheWord
           (isa ?oblique-object GeographicalThing))
 :base ((permanentlyEastOf ?oblique-object ?noun)
       (isa ?oblique-object GeographicalThing)
       (isa ?noun GeographicalThing)))
```

Extracting adjective-based linguistic information from ResearchCyc produced 12,561 rules for REDMAP.

### 3.3.2.4 Bad Rules

Although the rule extraction templates provided a tremendous amount of leverage, not all rules extracted were correct or useful. Automated testing performed by the extraction algorithm identified nearly 4,000 incorrect attempted rule instantiations. These errors were distributed non-uniformly throughout the output of 31 of the 64 extractors. There was not always a trivial mapping between the information represented in ResearchCyc assertions and REDMAP rules, and therefore assumptions had to be made in the rule extractors. Extractors were written to apply to the majority case of the information usable by REDMAP for the assertions they were mining. Some mappings were more straightforward than others were.
Nearly three quarters of the errors are attributable to the output of just three extractors. These extractors were associated with the predicates multiWordString, compoundString, and headMedialString. (See section 3.3.2.2 for additional discussion of these types of assertions and extractors.) These assertions map sequences of strings or lexical concepts to semantic and episodic concepts in ResearchCyc. It was incorrectly assumed in the extractor template that these predicates only mapped to collections, and thus the extractor’s output template for the base of the rules being produced was the following.

(\texttt{isa } ?x \texttt{<thing-mapped-to>})

The assertions in ResearchCyc for these three predicates frequently mapped to concepts such as named entities, SchongenTreaty, predicates, pulseRate, units of measurement, Kilocalorie, and ResearchCyc microtheories, MicrobiologicalChemistryMt, all of which are not correctly represented when substituted into that template, as they do not represent collections. These incorrect instantiations were filtered out. Even if they were correctly identified by an extractor some of the filtered assertions are still not useful to REDMAP. For example, having the lexicon populated with the proper names of all of ResearchCyc’s microtheories is of minimal utility. Mappings from text strings to predicates without knowing how to generate a pattern to assign arguments to those predicates would also be worthless. Therefore, many extraction errors were generated by assertions that would be difficult or impossible to transform automatically into REDMAP rules anyway.
Many other errors are attributable to bad assertions in ResearchCyc itself, for example specifying an output template for a rule that violates the semantics of the ResearchCyc ontology, or even the syntax of CycL. These errors included assertions in the semantic templates missing arguments, or violating their own type constraints. Other errors in ResearchCyc were manually identified, such as the assertions that “X in Y” means both \((\text{objectFoundInLocation} \; X \; Y)\) and \((\text{objectFoundInLocation} \; Y \; X)\). Obviously only one of these interpretations is correct.

In addition to incorrect rules, rules that mapped very general language to concepts so high in the ontology that they were effectively void of meaning were also removed. An example of this type of rule would be mapping the pattern “\(<\text{thing}> \text{ by } \langle\text{thing}\rangle” to the assertion \((\text{by-Underspecified} \; ?\text{thing1} \; ?\text{thing2})\). Such a rule would match both the “next to” and “authorship” senses of the word “by,” and map them to the same useless predicate.
Chapter 4 Reference Resolution

This dissertation focuses on the type of understanding necessary to learn from evolving descriptions of people, places, events, etc. described over multiple sentences and across multiple documents. This level of understanding requires the ability to resolve references to episodic memory within and across sentences.

Reference resolution is the process of identifying to what concepts in memory the language is referring by resolving how references are grounded to existing semantic and episodic knowledge. The reference resolution algorithm presented in this chapter handles all ambiguity with a uniform mechanism, resolving references not only between semantic structures, but also grounding those references to existing episodic knowledge structures when and if they already exist. Integration with the underlying memory is part of the reference resolution process, rather than yet another step in a pipeline model.

For each piece of text, multiple ambiguous interpretations will likely be produced by following the mappings in the lexicon and in the rules (described in chapter 3). Each of those ambiguous interpretations is made up of predicate logic representations which may or may not have all of their references grounded in existing episodic knowledge structures. Forming an understanding for a piece of text requires selecting the correct interpretation or set of interpretations from the set of ambiguous interpretations and fully grounding all references in episodic memory.
4.1 Reference Resolution Example

Consider the following input text.

T1
Egypt is in Africa.
Egypt borders Sudan on the south.
The country borders the Mediterranean Sea.

The output from pattern matching for the first sentence consists of the following assertion.

F1
(inRegion Egypt ContinentOfAfrica)

The second sentence contains an ambiguity. From the text alone it is not possible to accurately
determine which country is north of the other. The pattern matching output for the second
sentence of T1 includes the following two candidate interpretations.

F2a
(politiesBorderEachOther Egypt Sudan)
(permanentlyNorthOf Egypt Sudan)

F2b
(politiesBorderEachOther Egypt Sudan)
(permanentlyNorthOf Sudan Egypt)

Note that F2b represents the (incorrect) interpretation of the second sentence of T1 where Egypt
is south of Sudan. REDMAP must select which of these interpretations to merge with the
running interpretation. The correct interpretation can only be made by applying real-world, episodic knowledge. By querying for these two groups of assertions in its existing episodic memory, REDMAP can identify that the assertions in F2a are all already known. On the other hand, only one assertion in F2b is known, while the underlined assertion is not. Making these queries allows REDMAP to use its existing episodic memory to prefer F2a over F2b. The running understanding now consists of F1 and F2a.

Continuing with the processing of text T1, the pattern matching step in REDMAP will produce the following candidate interpretation for the third sentence.

\[
\text{F3} \\
(bordersOn-AgentAgnostic MediterraneanSea ?country) \\
(isa ?country Country)
\]

The interpretation of this sentence contains an ambiguity in the form of an ungrounded, yet to be identified, instance ?country. REDMAP needs to identify how to fit its interpretation of the third sentence, F3, in with the running interpretation, and identifying to what ?country refers. REDMAP will do this by first trying to connect the ungrounded reference with any of the existing references in the running interpretation, which is composed of F1 and F2a. In the running interpretation there are three existing references: Africa, Egypt, and Sudan. REDMAP will check connecting to these references first, before assuming that ?country refers to some completely new, fourth, reference. REDMAP first compares the existing type information it has about the references. Africa is not type-compatible with being an instance of the collection Country and it is ruled out as a possible referent for the ungrounded reference ?country. Egypt
and Sudan however, are both compatible references based on type information alone. These references are compatible with semantic constraints, so REDMAP will check them both against its episodic memory. It does this by substituting each candidate value into the assertions in F3 in place of the open reference, ?country. It will then query episodic memory for the following two possible grounded interpretations.

\[(\text{bordersOn-AgentAgnostic MediterraneanSea Egypt})\]

\[(\text{bordersOn-AgentAgnostic MediterraneanSea Sudan})\]

REDMAP will confirm that the first exists in episodic memory, while the second is unknown, causing the first interpretation to be preferred. Thus REDMAP is able to combine F3 with the running interpretation by identifying that ?country refers to Egypt, in doing so it has resolved the reference and produced an understanding that is fully grounded in existing episodic memory.

### 4.2 Forming Understandings

When text is ambiguous, REDMAP tracks multiple candidate interpretations simultaneously. A candidate interpretation consists of three things. The first is a set of interpretation fragments selected, using the algorithm described in the next subsection, from the results of pattern matching for each sentence of the text read thus far. Collectively the fragments contain all of the assertions in the interpretation. The second component of an interpretation is a set of reference groups that map references across interpretation fragments to each other. For example, this is where it would be recorded that a reference to a bombing event in the interpretation of one
sentence is the same as an attack event in the representation of another sentence. The third component of an interpretation is a set or sets of bindings that ground the reference groups to existing episodic instances. An interpretation can have multiple sets of bindings in the case that it is ambiguous with respect to the underlying memory and multiple existing episodes fit the interpretation constructed thus far. It is also possible for an interpretation to have no bindings, in the case that it represents completely new episodic knowledge. A complete understanding is formed by substituting a set of bindings in for their corresponding references in the assertions. When the interpretation is stored in memory as the final understanding of a text any reference groups without bindings will have new instances reified as their values. Figure 4.1 presents a depiction of how interpretations are organized.
REDMAP produces its understanding incrementally, and at any step can provide its best interpretation, or a set of ambiguous interpretations based on what it has read so far. REDMAP updates its understanding after each sentence. Understanding could be updated after smaller pieces of text are processed, such as phrases, or even after larger pieces of text, such as multiple sentences or paragraphs. Results in section 6.1 show that processing units smaller than sentences was computationally prohibitive. The patterns extracted to produce REDMAP rules provide understanding only up to the sentence level, and so this is the logical place to perform incremental updates of understanding. Interpretations are updated using the following algorithm.

### 4.2.1 Reference Resolution Algorithm

REDMAP starts extending its understanding of the text by first prioritizing its list of running interpretations using the following two criteria in order.

1. Preference is given to interpretations that have more references groups grounded via bindings to existing episodic memory.
2. Interpretations that represent a more interconnected set of assertions are preferred over those with unconnected assertions.

The first is a measure of how well a given interpretation is recognized. The second is a measure of the coherence of an interpretation and is discussed in detail in section 5.2.2. The most preferred interpretation is always selected to be extended.
The output of the pattern matcher for each sentence consists of multiple ambiguous sets of assertions, referred to as *interpretation fragments*. Interpretation fragments consist of one or more markers produced by the mappings in REDMAP rules (described in chapter 3). An interpretation fragment will contain multiple markers when the output of several rules is connected, as is the case in the nested pattern matching example in section 3.2.3. An interpretation fragment is selected by applying the following preferences in order.

1. Fragments that were produced by rules that matched more words of the input text are preferred over those that matched fewer.
2. Fragments containing assertions that use structures in the ontology, such as collections, that are already used in the running interpretation that was selected to be extended are preferred over those that do not.

After a running interpretation and an interpretation fragment are selected their representation are merged and evaluated. This process will either result in a new, combined, interpretation being produced, or for the pair to be discarded as incompatible, in which case another pair will be selected and evaluated.

New content is merged into an interpretation as follows. For each reference in the new assertions, a set of compatible reference groups in the running interpretation are identified as candidate referents. Each reference also has the option of not being connected to an existing reference group in the running interpretation, which occurs if the new reference is referring to a
concept not yet introduced in the running interpretation. All possible permutations of these candidate reference mappings are enumerated and prioritized using the following two criteria.

1. Preference is given to those mapping that connect more references to existing groups in the running interpretation over those that connect to fewer existing reference groups.
2. Mapping containing exact type matches are preferred, e.g., mapping a reference of type Bombing to a existing reference group of type Bombing, over those that are connected via generalization information.

The reference mappings are then evaluated in order until a preferred mapping is identified. A reference mapping is evaluated by iterating over all the bindings in the running interpretation. For each binding, its values are substituted into each new assertion using the reference mapping to translate between the new references and the reference group to which the bindings are associated. Each translated assertion is then compared with existing episodic memory. This comparison yields one of three values. First, the assertion could be already known in episodic memory, and it would be labeled confirmed. Second, the assertion could be unknown in existing memory, but not contradicted by anything known, in which case the assertion is consistent. The third option is that the assertion contradicts existing memory, and it is rejected, along with the set of binding currently being evaluated for this mapping. If the assertion is confirmed or consistent, evaluation will continue with the next transformed assertion. If all assertions for a set of bindings are confirmed or consistent, the set of bindings will be kept. If at least one set of bindings is kept, the mapping is selected as the way to merge the new assertions with the running
interpretation and a new interpretation is produced. In addition to checking if an assertion is confirmed or consistent, querying memory will also retrieve existing episodic instances for new references which did not exist in the running interpretation. In this way references are resolved and grounded to existing episodic memory simultaneously. Any values returned will be combined to extend the existing bindings in the new interpretation.

If an assertion is neither confirmed nor consistent, the set of bindings being evaluated is immediately rejected. If all bindings are rejected by a given mapping, that mapping will also be rejected and the next reference mapping will be evaluated. If all bindings are rejected for all mappings, or there are no bindings, it is assumed that the interpretation is representing something that is not already known in episodic memory. In this case the highest preferred reference mapping, based on type information alone, (which is the first one that was evaluated) is used to connect the new assertions to the running interpretation. This new interpretation will then have no bindings.

After a new interpretation is created, all candidate interpretations are re-prioritized, as at the beginning of the algorithm. If the new interpretation is not on top the next candidate interpretation is selected and extended in the same manner. In this way the algorithm can search for an understanding best-first, but backtrack and evaluate other options if a decision no longer looks good after being explored.
4.3 Examples of Leveraging Memory

The following subsections step through five major ways in which the reference resolution algorithm in REDMAP interacts with memory to resolve ambiguity in forming its understanding. Each example illustrates how existing knowledge in memory affects the decisions REDMAP makes.

4.3.1 Filtering by Shared Exemplars

In order to understand descriptions that span multiple sentences, REDMAP must be able to resolve references across those sentences. To do so REDMAP must be able to identify if two references can refer to the same concept. This problem can be formulated as identifying whether an instance of one collection could also be an instance of another collection. For example, in order to understand the following story REDMAP must identify that the event referred to in the first sentence is the same as that in the second.

T2
The bombing occurred in Iraq.
The attack killed 14 soldiers.

The text “attack” produces a reference to an instance of the collection AttackOnObject, and “bombing” produces a reference to an instance of the collection Bombing. It can be trivially known that an instance of Bombing can be an instance of AttackOnObject, because one is a generalization of the other in the ontology. The filtering of candidate reference mappings based
on type information performed in the reference resolution algorithm will allow this option to be investigated.

However generalization relations in the ontology are not enough to allow all acceptable reference mappings. In the following example it is again necessary, in order to understand the text, to know that the events referred to in each sentence are the same event.

**T3**

The explosion occurred in Iraq.
The attack killed 12 soldiers.

In this example there is a reference to an instance of Explosion and an instance of AttackOnObject. In this example, unlike T2, there is no generalization relation between Explosion and AttackOnObject as not all explosions are attacks (*e.g.*, mining explosions, supernovae, etc.). If type filtering only allowed a candidate mapping if they were the same type, or if one was a generalization of the other, this text could not be properly understood.

Complicated reasoning based on semantic information could be performed to try to rule in this mapping, however because REDMAP has access to episodic memory a simpler solution is available. Since the question to allow a mapping can be posed as, “can an instance of A be an instance of B?” REDMAP can query episodic memory for an example of such an instance. REDMAP can simply query its existing episodic memory for an instance of an attack that is also an instance of explosion, effectively asking, “Have I heard of one of these before?” This turns what could be a complicated reasoning task about the relationship between two types into a
simple lookup in episodic memory. However, this technique is obviously limited in sparse sections of the knowledge base where such an instance may not already be represented.

4.3.2 Incremental Reference Resolution

The following examples illustrate how REDMAP uses episodic memory to resolve references incrementally across sentences and to existing episodic memory structures. These examples illustrate how bindings are initially retrieved from episodic memory, how they are used to make reference mapping decisions, and how they are then extended in order to ground new references.

T4.1 Two buildings were destroyed in New York.

Starting with the preceding sentence, T4.1, REDMAP will produce the following semantic structure.

```
F4
(isa ?destroy DestructionEvent)
(eventOccursAt ?destroy CityOfNewYorkNY)
(inputsDestroyed ?destroy ?buildings)
(isa ?buildings (GroupFn Building))
(groupCardinality ?buildings 2)
```

This interpretation contains two references that are not yet grounded in episodic memory: a destruction event, ?destroy, and a group of buildings, ?buildings. REDMAP will query memory for that structure, to attempt to ground those references to existing episodic memory structures. In doing so it will identify the instance September11AttackNewYorkCity as a value

In addition to F4, pattern matching will result in additional candidate interpretations for T4.1 that correspond to the meanings of these subsets of the text: “Two buildings… in New York,” “Two buildings were destroyed…,” and “Two buildings were… in [popular]….” However, the interpretation that maps to F4 will be strongly preferred because it refers to existing instances in memory, and matches more text than any of the other, shorter, partial understandings that may retrieve instances as well.

After reading the first sentence and producing a candidate running interpretation, REDMAP will process the next sentence.

T4.2a The attack was performed by Al-Qaida.

Sentence T4.2a produces the following candidate interpretation.

F5a
(isa ?attack AttackOnObject)
(performedBy ?attack Al-Qaida)

REDMAP must now decide how to add this to its running understanding of the story. REDMAP needs to find possible reference mappings between the destruction event, the city of New York, and a group of buildings in the first sentence and the attack event, and the terrorist group Al-Qaida in the second sentence. In this case after type filtering, there is one possible mapping, that
the attack could be mapped to the destruction. Rather than performing inference on general concepts to evaluate this mapping, REDMAP asks memory if the specific instance it has for the destruction event \texttt{September11AttackNewYorkCity} was or could be an attack by Al-Qaida by substituting the value in and issuing a query to memory for the following assertions.

\begin{verbatim}
(isa September11AttackNewYorkCity AttackOnObject)
(performedBy September11AttackNewYorkCity Al-Qaida)
\end{verbatim}

Indentifying these assertions in memory reinforces the reminding to the existing memory structure \texttt{September11AttackNewYorkCity}, and provides evidence for merging the references \texttt{?attack} and \texttt{?destroy}. REDMAP has simultaneously resolved the reference and grounded it in memory. In lieu of subsequent contradictory information REDMAP will continue to believe this interpretation is correct.

If instead the example is changed slightly, so the second sentence is the following.

\textbf{T4.2b} The attack was performed by terrorists.

REDMAP will behave in the same fashion, however, the candidate interpretation of the second sentence will be the following.

\begin{verbatim}
F5b
(isa ?attack AttackOnObject)
(performedBy ?attack ?terrorist)
(isa ?terrorist TerroristGroup)
\end{verbatim}
Note this time it does not yet know that the attack was performed by Al-Qaid, only that some yet to be identified instance of the collection TerroristGroup performed the attack. In addition to having to resolve the reference for the attack, ?attack, as before, REDMAP will now also have to resolve the reference to the terrorist group, ?terrorist.

After performing type filtering the only option for mapping references is to connect the attack to the destruction event. As before, REDMAP will ask if ?attack can be merged with ?destroy, by substituting known values into the set of assertions, and querying memory. This time the query is being issued with the performer being an ungrounded reference.

\[(\text{isa} \; \text{September11AttackNewYorkCity} \; \text{AttackOnObject})\]
\[(\text{performedBy} \; \text{September11AttackNewYorkCity} \; ?\text{terrorist})\]
\[(\text{isa} \; ?\text{terrorist} \; \text{TerroristGroup})\]

With the same assertions in memory as before, this query will still return true, again reaffirming REDMAP’s decision to merge the two reference, furthermore the response from memory will indicate that the previously open reference ?terrorist should refer to the known entity Al-Qaida. This allows REDMAP to ground that reference to a known episodic memory structures in the same step.

REDMAP uses an open world assumption when interacting with its memory. Changing the example one more time, if the assertion about Al-Qaida performing September11AttackNewYorkCity did not already exist in memory after reading the first version, sentence T4.2a, REDMAP would see that it does not know who performed the event
REDMAP would not be able to identify the assertion in memory, instead of being confirmed, this assertion would only be marked consistent. Barring contradictory information, REDMAP will ultimately assert that Al-Qaida was the performer, since that is what the text proposed. If however the performer is unknown to memory and the second version, sentence T4.2b, is read, REDMAP will not be able to identify the performer, as neither the text nor memory ground the reference to an existing episodic structure. This will cause REDMAP to ultimately reify a new instance for ?terrorist to store in memory, since neither the text nor memory references a known entity.

4.3.3 Reference Resolution with Multiple Remindings

In addition to the text being ambiguous, remindings to existing episodic knowledge could also be ambiguous. This can result in multiple remindings being retrieved from memory and an interpretation having multiple sets of bindings. This is especially true of a text that starts with a very general description, for example T5. When multiple existing structures are retrieved they cannot all be what the text is referring to, and potentially none will be correct, in the case where REDMAP is seeing a story for the first time.

T5
An attack occurred in Yemen.
The attack killed 17 sailors.
The attack injured 39 sailors.
The attack occurred on October 12, 2000.

REDMAP performs incremental understanding and starts grounding references in existing knowledge structures as soon as it reads the first sentence. The following example illustrates
how an ambiguity that results in multiple bindings is resolved over the course of processing several sentences.

**T5.1** An attack occurred in Yemen.

Starting with the preceding sentence, T5.1, REDMAP will retrieve 17 candidate events from memory that this sentence could be describing. The underlying knowledge base knows about a lot of attacks that occurred in Yemen that occurred on several different dates. The following are three examples.

\begin{itemize}
  \item \texttt{TerroristAttack-January-21-1998-Yemen}
  \item \texttt{TerroristAttack-July-16-1999-Yemen}
  \item \texttt{TerroristAttack-October-12-2000-Yemen}
\end{itemize}

Reading the next sentence adds additional constraints to the running interpretation.

**T5.2** The attack killed 17 sailors.

The first reference mapping evaluated will propose connecting the reference to an attack in the second sentence to the attack in the running interpretation. REDMAP will substitute each of the 17 remindings produced by the interpretation of the first sentence into the corresponding reference in the second sentence. The mapping between the two attack references will be preferred if not all of the remindings are ruled out. REDMAP will keep a reminding if, in this case it can confirm in memory that the particular event killed 17 sailors. It will also keep an
event where it does not know how many sailors were killed, as it is still consistent with the text and interpretation. However, if one of the events is known to have killed a different number of sailors, then its reminding will be discarded as it is inconsistent with the information being provided in the text. In this case four events from the original 17 could be ruled out, since REDMAP could identify they were inconsistent with killing 17 sailors. REDMAP is still left considering 13 candidates.

Understanding continues by adding the interpretation from the following, third, sentence to the running interpretation.

T5.3 The attack injured 39 sailors.

As with sentence T5.2, REDMAP needs to identify that that attack referred to in sentence T5.3 is the same as the one in the running interpretation. It will again do this by substituting all the bindings it is tracking for the running interpretation into the reference in the new interpretation and look for them in memory. All 13 events are at least consistent with this additional knowledge, and therefore REDMAP can not rule out any of them. The fourth sentence contains more information.

T5.4 The attack occurred on October 12, 2000.
From the information in this sentence REDMAP is able to rule out 12 of the remaining 13 events, as well as confirm that the last remaining event is known in the knowledge base to have occurred on the date mentioned in the text.

Now that REDMAP has reached the end of the text, it has identified that the story is discussing TerroristAttack-October-12-2000-Yemen, and it knows the information contained in following assertions already exists in the knowledge base.

(eventOccursAt TerroristAttack-October-12-2000-Yemen Yemen)
(dateOfEvent TerroristAttack-October-12-2000-Yemen
     (DayFn 12 (MonthFn October (YearFn 2000)))))

At the same time REDMAP was building its semantic representation of the story it was also identifying how it is connected to existing knowledge. No additional time is needed to ground this representation in the knowledge base. In addition to the existing assertions just listed, some assertions in the final understanding do not already exist in memory. They were identified as being only consistent with what exists in memory. In completing its understanding of the story REDMAP proposes that the following two assertions be added to the representation of TerroristAttack-October-12-2000-Yemen, extending the existing episodic structure.

(deathToll TerroristAttack-October-12-2000-Yemen
     CrewMemberOnShip 17)
(injuryCount TerroristAttack-October-12-2000-Yemen
    CrewMemberOnShip 39)

To complete the integration of the story with its existing knowledge, only these two assertions need to be added to the knowledge base.
4.3.4 Conflicting Role Fillers

Understanding of all of the examples presented thus far required REDMAP to merge references across sentences. In fact in the algorithm presented, REDMAP is biased to merge as many references together as it possibly can in order to produce the most coherent interpretation possible. However, in contrast to example T2, where understanding required merging the reference to a bombing in the first sentence to an attack in the second, understanding the following example requires knowing that the two events cannot be the same.

\[
\text{T6}
\]

Israel bombed the Beirut International Airport.
Hezbollah attacked Haifa.

The interpretation of the first sentence will include the following assertions.

\[
\text{F6}
\]

(\text{isa} \ ?\text{bombing} \ \text{Bombing})
(\text{performedBy} \ ?\text{bombing} \ \text{Israel})

The interpretation for the second sentence will include these assertions.

\[
\text{F7}
\]

(\text{isa} \ ?\text{attack} \ \text{AttackOnObject})
(\text{performedBy} \ ?\text{attack} \ \text{Hezbollah})

Although REDMAP is biased to attempt to connect \text{?bombing} and \text{?attack}, a correct understanding is formed only by knowing that they must be different events. REDMAP can make this decision by identifying that the bombing has its \text{performedBy} role filled by Israel, and the attack’s performer is Hezbollah. Thus when it attempts to merge the two references, it will
see that running interpretation wants to fill the new combined event’s `performedBy` role with a value representing Israel, while the new assertions in F7 want to fill this same role with a value representing Hezbollah. Further needed to make this decision is the knowledge that the entities Israel and Hezbollah are neither equivalent nor part of a metonymic relationship, or some other relationship which would allow them to cooperate in this role. REDMAP does not perform any reasoning when deciding if two role fillers conflict. It only allows a mapping if the fillers are equal, or in the case of general references e.g., soldiers or people, if they are type compatible using the methods described in section 4.3.1. This works here but is an obvious place for future improvement.

### 4.3.5 Semantic Expectations

Ambiguous references can also be resolved by understanding how instances of a given type are typically structured in memory. For example, consider the following story, in which the third sentence has been intentionally made vague to more clearly illustrate this point through the use of the word “event”, instead of something more specific like “bombing” or “attack”.

```
T7
Hezbollah kidnapped 2 Israeli soldiers.
Israel bombed the Beirut International Airport in response to the kidnapping.
The event damaged 3 runways.
```

A human can trivially understand this story, and resolve the reference to the event in the third sentence to the bombing mentioned in the second sentence. There are two clues which can be
used to resolve this ambiguity. First there is the relationship between airports and runways. Second there is an expectation that bombings damage things, but kidnappings might not.

Given everything described thus far about the way REDMAP resolves references, it has no particular reason to prefer merging the event in the third sentence with the bombing over merging it with the kidnapping. However, in addition to the type filtering detailed earlier in this chapter, section 4.3.1, REDMAP also looks at the resulting semantic structures that will be produced if it merges two references, and uses existing semantic knowledge to prefer one structure over another.

For example, after processing the first two sentences of T7 the following representations are constructed.

F6
(isa ?KidnappingSomeone-1 KidnappingSomeone)
(perpetrator ?KidnappingSomeone-1 Hezbollah)
(agentCaptured ?KidnappingSomeone-1 ?Soldier-Group-2)
(isa ?Soldier-Group-2 (GroupFn Soldier))
(groupCardinality ?Soldier-Group-2 2)

F7
(isa ?Bombing-3 Bombing)
(objectHarmed ?Bombing-3 BeirutAirport)
(performedBy ?Bombing-3 Israel)
(inReactionTo ?Bombing-3 ?KidnappingSomeone-1)

Upon reading the third sentence the following assertion structure is produced.
Because the third sentence is vague, \( ?\text{Event-4} \) could refer to either the bombing or the kidnapping. There is no reliable heuristic which can resolve this ambiguity without leveraging semantic knowledge. A heuristic which says use the most recently mentioned type-applicable referent will fail, as the most recently mentioned referent is the kidnapping. Location in the type ontology provides no help either, as both possible referents are effectively equidistant from the type \textit{Event}, in contrast to using a different reference such as “airstrike” which is a type of bombing and also closer to bombing than kidnapping in the ontology.

To merge F8 with the running interpretation composed of F6 and F7, there are two type-applicable mappings for the reference \( ?\text{Event-4} \). It could be mapped to either \( ?\text{KidnappingSomeone-1} \) or \( ?\text{Bombing-3} \). REDMAP resolves this ambiguity by first identifying the set of predicates in the new set of assertions that take the reference as an argument. In F8, \( ?\text{Event-4} \) occurs in an assertion with the predicate \textit{damages}. If this reference is merged with another existing reference, the new combined reference will then be used as an argument in a \textit{damages} assertion.

REDMAP prefers mapping references to reference groups whose referents are more likely to be an argument for a particular predicate than those that do not. For example, ResearchCyc has higher-order assertions that indicate that instances of \textit{AttackOnObject} are expected to be an
argument in a damages assertion, because all attacks are likely to damage something. (See section 2.1.4 for a discussion of higher-order assertions in ResearchCyc that are a source of this information.) In this particular case Bombing is a specialization of AttackOnObject and thus its instances are expected to appear in a damages assertion. The same expectation does not exist for instances of KidnappingSomeone. Therefore REDMAP will prefer combining ?Event-4 with the reference group in the interpretation that is associated with ?Bombing-3.

Finishing this example, no applicable reference can be found for the group of runways, and therefore REDMAP assumes it represents new information. When stored in the knowledge base a new instance will be reified to represent the group of runways.

### 4.4 Challenges in Interacting with Memory

REDMAP interacts with knowledge using version 2 of the FIRE reasoning engine (Forbus and De Kleer 1993). The knowledge base used was extracted from ResearchCyc 1.0. These choices in technologies were dictated primary by the goals and constraints of the Learning Reader project. Along with these decisions came several challenges for DMAP, and REDMAP specifically, described in the remainder of this section.

#### 4.4.1 Knowledge Organization

Unlike memories used with previous DMAP implementations, ResearchCyc is not frame-based. Frame-based systems frequently provide high-level structures which represent generalizations
and relationships between the components of those generalizations, such as the information encoded in scripts describing relations between sequences of events, for example, dining at a restaurant. Although the ResearchCyc ontology has the ability to represent script-level detail, as well as many other forms of generalization, this type of knowledge is far under-represented in ResearchCyc. As a result, building larger knowledge structures and performing reference resolution was a significantly larger and more complicated task for REDMAP as compared to previous DMAP research, requiring a radical re-envisioning of the core DMAP processes.

Just as collections can represent generalizations of other collections, predicates also have a generalization hierarchy in ResearchCyc. For example, performedBy is a specific type of the predicate doneBy, the primary distinction being that performedBy indicates that the actor performed the action intentionally, while doneBy does not ascribe intentionality. An actor accidentally knocking over a glass would be represented with doneBy, while throwing a plate at someone would be more completely represented with performedBy. Absent from ResearchCyc is a predicate for an event being represented as being performed unintentionally by a given agent. It is therefore not possible to tell whether an event represented with doneBy is fully represented, or it is merely unknown whether a more specific representation holds. This problem appears in multiple locations in both the predicate hierarchy and in the collection hierarchy in ResearchCyc. REDMAP leaves open the possibility that assertions and knowledge in ResearchCyc is underrepresented in this fashion and allows more specific content to match to less specific content.
In contrast to the previous assumption, REDMAP makes a slightly different assumption when it comes to the open-world problem. It is possible for knowledge to be incompletely represented in ResearchCyc. Incomplete descriptions could come in two types. The first is that content could be completely absent, for example, a bombing not having an attached \textit{performedBy} assertion. The second is that only one of a set of performers could be represented in the knowledgebase. When knowledge is completely absent, for example, no \textit{performedBy} role exists for an event, REDMAP entertains an open-world hypothesis and assumes the performer is unknown to the knowledge base. REDMAP will view any performer for this event, mentioned in the text, as being consistent with that event being performed by the given agent. However, if \textit{performedBy} roles for a given event exist, REDMAP assumes that all \textit{performedBy} information for this event is specified. REDMAP treats unrepresented knowledge as unknown, while represented knowledge is assumed to be complete.

ResearchCyc segments its knowledge into microtheories in order to attempt to encapsulate inconsistencies (see section 2.1.5). One microtheory could contain knowledge that is incompatible with another microtheory. It was noted in section 2.1.5 that REDMAP performs its operations in the context of the microtheory \textit{EverythingPSC}, which could contain inconsistencies. This is very efficient for REDMAP since it does not have to query into multiple, varying contexts to cover the entire knowledge base. REDMAP maintains relative immunity from potential conflicts due to inconsistencies since it is not actually performing any reasoning in this context. REDMAP’s primary operations are to confirm that a certain assertion already exists in memory, and to retrieve sets of assertions that conform to a given structure. If there are
inconsistent interpretations, REDMAP will retrieve them both and track them independently. Since REDMAP is retrieving existing knowledge it should be unlikely for it to combine inconsistent assertions in the same interpretation if they are not already connected in the knowledge base. It is an open question as to how unlikely this is. The interpretations produced by REDMAP do contain a list of microtheories that assertions were drawn from allowing subsequent reasoning to be performed to check for conflicts. However, one microtheory being inconsistent with another does not mean that any given assertion in one is guaranteed to conflict with one or more assertions from the other. Reasoning for inconsistencies is nontrivial and beyond the scope of this research.

4.4.2 Engineering Limitations

The massive amount of knowledge in ResearchCyc and the methods REDMAP uses to interact with it raise issues of scale. These issues needed to be addressed aggressively in REDMAP. This section presents a discussion of some of the approaches taken to reign in processing time. Scalability is a major component of this work and is evaluated in chapter 6. While the algorithms described earlier in this chapter have exponential worst-case running times, chapter 6 will show that this concern was unsubstantiated for the evaluation corpus given the following optimizations.

One of the most frequent operations REDMAP performs is to check if a reference to a given collection is type-compatible with a reference to another collection, or list of collections. This is primarily done by querying if there is a generalization relation between the two collections, using
the predicate genls. FIRE has optimizations for querying for this relationship, however profiling an early implementation of REDMAP showed 60% of processing time was still being spent computing generalization relations. Since many similar generalizations are queried repeatedly, REDMAP hashes the results from FIRE for each query and looks for results in this table before querying FIRE, mitigating the cost of multiple requests for the same information.

Retrieving bindings from memory requires REDMAP to query for the partially formed assertions mapped to by rules. The structure of these sets of assertions is specified in the templates extracted from ResearchCyc. Some of these templates can be quite inefficient in either what they are querying for or in the ordering of their conjuncts. Furthermore, poor matches by REDMAP can compound the problem. For example, if the pattern matcher sees “<event> performed by <agent>” with no constraints, this could result in a query for all events performed by anyone. Processing time in FIRE for a conjunct is order dependent. FIRE will retrieve all bindings for the first branch, and then proceed to the second, etc. So a query requesting all the bombings performed by terrorists can run in a wide range of times. For example the following will be quite slow, as it identifies all terrorists, then generates the cross product of all terrorists and all bombings, and then decides if any of those pairings exist in a performedBy relation.

(and (isa ?agent TerroristGroup) (isa ?event Bombing) (performedBy ?event ?agent))

REDMAP contains a query optimizer that performs a static analysis on any query to first to remove constant or dead branches. For example, if REDMAP detects a constant branch that will
always return false in a conjunct, the entire query can be aborted as a failure without running it. It will also reorder branches to attempt to minimize the number of intermediate bindings that will be produced and investigated. The previous example will be optimized to the following.

```lisp
(and (isa ?event Bombing)
     (performedBy ?event ?agent)
     (isa ?agent TerroristGroup))
```

While this type of optimization does not come without overhead, it provided a significant reduction in REDMAP running time. For the Lebanon corpus this reduced overall running time by 26%.

Since REDMAP will evaluate multiple alternatives, some of which very similar, further query optimization is possible. After queries are optimized, their variable names are normalized and the results are cached. The results are kept only for the processing of a given text. This further improved processing time when REDMAP must backtrack or when it repeatedly generated overly general queries. While in general these optimizations and the fact that as more text is read REDMAP is able to hone in on what is being discussed, as opposed to exploring more and more ambiguity with each sentence, lead to the algorithms in REDMAP being tractable, problem cases are still encountered. These are resolved by implementing two thresholds on processing complexity. This first is a timeout on queries. In general the kinds of querying REDMAP is issuing should ideally either fail fast or return a small set of candidates. REDMAP will assume failure on a query it cannot process under threshold. The thresholds are currently set quite high, 30 seconds
per word in pattern matching, and one minute per binding retrieval operation. The second threshold is for when too much ambiguity is produced by the pattern matcher. Ideally REDMAP should explore the semantic and episodic fit of each alternative. When there are more than 24 options that cover the same span of text, REDMAP will reduce the set by selecting the 4 that involve the predicates and collections that occur most frequently with those in the running interpretation.

4.5 Discussion

This chapter presented the way REDMAP formulates understanding of a text, and how it incrementally resolves references by grounding them to existing episodic structures. This approach performs language understanding by viewing the problem as one of recognition. Integration with existing memory is performed as part of the process of resolving references. This is in stark contrast to traditional bottom-up approaches that first construct a semantic meaning and resolve references, and then attempt to integrate with memory.

Since ambiguity can be deferred, and this approach also introduces episodic ambiguity, scalability is an appropriate concern. This is a primary topic of this dissertation and the evidence for claims supporting the scalability of this approach is discussed in detail in chapter 6.

The current reference resolution algorithm produces candidate referents for all new references, and then produces and evaluates complete mappings that have a referent for each reference. However, it could be more efficient to evaluate partial mappings first, before constructing the
complete mapping. For example, consider a running interpretation that has four reference groups representing a kidnapping, a bombing, a terrorist group, and a group of soldiers. If the next sentence refers to an event and an agent, there are four possible reference mappings, or six if one counts that both references could be to new information. If the new event is paired with the bombing, and the agent with the terrorists, and this mapping fails, REDMAP could then try pairing the new event again with the bombing, and the agent with the soldiers. However, if it was the connection between the new event and the bombing that was causing the mappings to fail this is being evaluated redundantly. Evaluating partial mappings, for example, by looking at the new assertions that contain only a single reference first, could make reference resolution far more efficient.

Memory-based approaches that use examples to drive their decisions can also be limited when examples do not exist in memory, or when novel concepts are being encountered. For example, the type filtering approach based on shared exemplars would not be able to allow the mapping between an explosion and an attack, if it had never seen one before. REDMAP can get around that limitation to some degree depending on how a text is written. For example, although REDMAP would be disinclined to merge references to Sunnis and insurgents, as ResearchCyc has no existing example of this, it will do so if the text explicitly refers to such a mapping, for example, “The insurgents were Sunnis.” In this case REDMAP will produce one reference that is a member of both the insurgent and Sunni collections. Referring to a known named entity as a member of multiple collections in separate sentences has the same effect as well.
Chapter 5 Coverage and Quality

The Learning Reader project set out to interact with and extend as much of the knowledge expressible in the underlying knowledge base as possible. In addition to storing the knowledge acquired from reading in the knowledge base, it is also used as input for an offline thinking process, the Ruminator. The Ruminator asks itself questions about what has been read, and also compares new information to existing knowledge via analogy. In addition to reading and ruminating, the Learning Reader must provide access to what it has learned via a question answering system.

Given this context it is essential for the output of the reading process to be tightly integrated with the underlying knowledge base. It is also important for understandings produced by reading to provide quality knowledge for question answering as well as facilitate the general reasoning performed by the Ruminator. This chapter evaluates the implementation of DMAP described in this dissertation, REDMAP, which is used to perform reading for the Learning Reader. Its coverage of the underlying knowledge base is evaluated, as well as the quality of the understandings produced. Specifically this chapter provides evidence for the claim that REDMAP can scale to large memories, and produce accurate and useful understandings. It also provides evidence for a weaker claim that the quality and quantity of assertions in interpretations can scale with the amount of language understood.

In addition to evaluating coverage of the underlying knowledge base, the quality of the understandings produced by REDMAP has been measured in four ways. First, every assertion
produced by REDMAP was identified as correct or incorrect, providing a metric for the quality of the assertions being produced. The output for each text was also evaluated with respect to a metric for interpretation coherence. However, just because an assertion is correct or comes from a coherent set of assertions, does not mean it is useful. Two more evaluations are presented that speak to the usefulness of REDMAP-produced knowledge. To do this, the effects of the knowledge produced by REDMAP on an automated question answering system were measured. Finally, the quality of deductive reasoning performed by the Ruminator, starting with the output of reading, is evaluated on the same question answering task.

While the work presented in this dissertation forms one of the largest evaluations of a DMAP system to date with respect to scalability, it is important to note the following limitations. The computation of knowledge base coverage represents only an upper bound on the recall of REDMAP. The evaluations in this chapter focus primarily on precision, and the measurements of assertion quality measured only precision (whether produced assertions were correct) not recall (whether all assertions that should have been returned were). Further while the focus was on precision, the corpus used is approximately 1,200 sentences covering nine domains, and thus represent only a small sample of the types of information that could be represented in the underlying knowledge base. Performing more exhaustive evaluations, especially for recall, would be extremely labor and time intensive and are beyond the scope of this dissertation, but are a source of ample future work.
5.1 Knowledge Base Coverage

The Learning Reader is intended to learn a wide range of conceptual knowledge. It must acquire knowledge about new instances as well extend existing descriptions already in the knowledge base. It must add new knowledge in such a way that systems capable of using the original knowledge would find the new knowledge accessible and useful as well. To that end it is important to understand how much of ResearchCyc (the underlying knowledge base) REDMAP is capable of interacting with. Learning Reader, and therefore REDMAP in this context, was not concerned with reading every possible way a given concept could be expressed. Since input could be simplified first, the primary goal was to understand at least one way any given concept could be expressed. The focus is on knowledge base coverage, and not language coverage.

A static analysis of REDMAP’s coverage of the knowledge base was performed to establish an upper bound on what concepts were reachable in English using REDMAP. This was done by identifying all of the collections and predicates referred to in the REDMAP lexicon and rules. (For a discussion of the organization of ResearchCyc and the terminology used see section 2.1. For a discussion of the knowledge extracted from ResearchCyc to populate the REDMAP lexicon, and to create rules, see section 3.3.)

The first step was to identify all of the collections referred to either in the lexicon or in the base set of a rule. These are collections that REDMAP can directly refer to in its output and reify new instances into. This set represents 57% of all the collection in the ResearchCyc ontology. REDMAP is also able to access collections, and instances of collections, to which it may not
have text patterns explicitly mapped by leveraging generalization relations (genls) in the ontology. For example, BoatBombing is a specialization of AttackOnObject, whereby a watercraft is used to deliver a bomb. There is no linguistic information in ResearchCyc associated with BoatBombing, but if an existing instance of this collection was described using more general language, such as “an attack,” REDMAP would be able to retrieve it. Although 57% of collections have language patterns attached to them, REDMAP can reach 99% (27,453 / 27,649) of the collections in the ResearchCyc ontology. However REDMAP is unable to produce new instances of collection without linguistic information directly associated to them. (See the following section for discussion of refinement, and how to overcome this limitation.)

With respect to relations in the ResearchCyc ontology, REDMAP has phrasal patterns that can produce and access 13% (1,175 / 9,191) of all predicates. There are 6,753 predicates in the ResearchCyc that are not used in any of the assertions in the knowledge base. (Another 455 are only used in one assertion.) Factoring unused predicates out, the number of predicates REDMAP covers increases to 15% (376/2438).

<table>
<thead>
<tr>
<th>Collections</th>
<th>REDMAP can Retrieve</th>
<th>REDMAP can Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>27,649</td>
<td>99%</td>
<td>57%</td>
</tr>
<tr>
<td>Predicates</td>
<td>9,191</td>
<td>13%</td>
</tr>
<tr>
<td>Assertions</td>
<td>1.28 M</td>
<td>43%</td>
</tr>
<tr>
<td>non-isa Assertions</td>
<td>0.98 M</td>
<td>29%</td>
</tr>
</tbody>
</table>

Table 5.1 Collections, predicates, and assertions in ResearchCyc accessible by REDMAP.
In terms of assertions, REDMAP can access 43% of the 1.2 million assertions in ResearchCyc, including \textit{isa} assertions. REDMAP can access 29% of the existing non-\textit{isa} assertions. Knowledge base coverage is summarized in table 5.1.

5.1.1 Discussion

Some of these numbers may seem low, for example only 13% of predicates are covered, however the distribution of how the predicates are used is heavily skewed. The \textit{isa} relation alone accounts for 23% of the assertions made in ResearchCyc (although those assertions are better discussed in terms of collection coverage). The 13% of predicates that REDMAP covers account for 29% of the total non-\textit{isa} assertions.

Since language and knowledge coverage is clearly non-uniform, it is important to look at how the predicates that are covered are distributed. The focus of this work is on reading, recognizing, and extending instance-based knowledge. To that end evaluating REDMAP’s coverage relative to the existing instances in ResearchCyc provides a metric for measuring this goal. Some predicates are only used with instances of some collections, and the predicates covered are not uniformly distributed across all collections. The predicate coverage for a collection can be computed by first identifying all the instances of the collection, and then identifying all the predicates of assertions where those instances appear as arguments. The predicate coverage for that collection is measured as the number of unique predicates from that set that REDMAP can access. There are 6,704 collections in ResearchCyc that have instances associated with them. Of those 4,480 (67%) have predicates that are covered by REDMAP. The mean predicate coverage for a collection is 41% and the median is 39%. The distribution is linear across those collections,
with 184 having 100% of their predicates covered, and 1532 having above 50% covered. Overall this coverage of the predicates associated with existing instance-based knowledge is well above the baseline predicate coverage of 13%.

REDMAP currently has no facility for performing what was is called refinement in the original DMAP work (Martin 1990). Refinement in DMAP uses specific arguments to general predicates to identify more specific predicates. For example, if in a bombing event the device used to deliver the bomb is a boat, the bombing could be reclassified as a BoatBombing. In REDMAP 99% of collections are either referred to by language, or connected via generalization information to those that are. If a refinement algorithm were implemented the number of collections that REDMAP could create new assertions for should increase to 99%.

### 5.2 Accuracy of Understanding

It is important to evaluate the quality of the understandings being produced by REDMAP. This provides not only a metric for REDMAP’s performance but is also useful for understanding the effect REDMAP is having on the underlying knowledge base, and by extension other processes that interact with the knowledge base. An evaluation was performed by having REDMAP produce understandings for the entire extended corpus, described in section 2.2. Two metrics, described in the following subsections, were used for evaluating the quality of the understandings produced.
5.2.1 Assertion Accuracy

The understandings produced by REDMAP for each text were evaluated for accuracy. All evaluation was performed by the author. The individual terms as well as for each assertion in the interpretation were evaluated. Each term was marked as either being relevant or irrelevant to the meaning being expressed in the text. A common source of irrelevant terms is selecting the wrong sense of a word. For example, consider the following text.

T1
Israel dropped 23 tons of bombs on a bunker.
The attack occurred on July 20, 2006
Israel said Hezbollah leaders were in the bunker.
Hezbollah said none of its leaders were in the bunker.
Israel has rejected calls for a cease fire.

The word “bunker” in ResearchCyc denotes both the concept Bunker, a fortified military facility, and the concept SandTrap, an obstacle found on a golf course bearing the same name. If in the interpretation of T1 the concept SandTrap appears it would be marked irrelevant, as that is clearly not a concept referred to by the text.

Each assertion was graded in a similar fashion. An assertion was marked correct if it is relevant to the interpretation of the text. Assertions were marked irrelevant if they were incorrect or if they represented content not presented in the text. For example the following two assertions would be marked as irrelevant.
The first assertion is irrelevant to T1, and the second is incorrect: Israel performed the bombing being discussed. An assertion that contains an irrelevant term is automatically marked as being irrelevant.

Evaluating REDMAP’s understandings in this way places every assertion into one of three categories: relevant, irrelevant, and irrelevant containing irrelevant terms. The third category is a subset of the second. This was done to be able to distinguish between errors made because REDMAP used the wrong building blocks, versus errors from using correct building blocks in the wrong places. The following examples represent a few assertions selected from REDMAP’s understanding of T1 to illustrate these three categories.

F1
(doneBy Dropping-1 Israel) correct
(primaryObjectMoving Dropping-1 GroupBomb-2) correct
(isa GroupBomb-2 Group) correct
(isa GroupBomb-2 (GroupFn Bomb)) correct
(groupCardinality GroupBomb-2 23) irrelevant

F2
(isa Speaking-3 Speaking) correct
(performedBy Speaking-3 Israel) correct

F3
(isa SandTrap-5 SandTrap) irrelevant term, irrelevant
(isa GroupLeader-6 (GroupFn Leader)) correct
(objectFoundInLocation GroupLeader-6 SandTrap-5) correct
The first four assertions in F1 are correct, however the last assertion is marked irrelevant. That assertion represents that the group of bombs Israel dropped was made up of 23 bombs. The text describes “23 tons of bombs” but makes no commitment to the actual number of bombs dropped. All of the assertions depicted in F2 are correct. The assertions in F3 are a little more complicated. They are clearly trying to represent a group of leaders located in a bunker. However, REDMAP has mistranslated “bunker” to the concept SandTrap, an obstacle found on a golf course. This concept has nothing to do with the text presented, and therefore the concept SandTrap is flagged as irrelevant. This also marks the first assertion in F3 as irrelevant. However, the remaining two assertions in F3 are correct. The concept SandTrap-5 is a REDMAP created entity. Without the first isa which is marked incorrect, it effectively becomes an ungrounded reference. The third assertion of F3 represents that the leaders are found somewhere, which is correct. It was only the assertion that described the details of that location (the first assertion of F3) that was incorrect.

Over all inputs, 72% of the assertions produced by REDMAP were correct. Due to the corpus being constructed in two stages, evaluation was performed separately for the core domains corpus and the Lebanon corpus. Of the 336 texts in the core domains corpus, 192 (57%) had interpretations with no irrelevant assertions, while only 14 (4%) contained no correct assertions. Furthermore, 237 (71%) of the texts had zero terms marked as not belonging in the interpretation. The results for the Lebanon corpus mirror those of the core domains corpus.
Table 5.2 Number of assertions produced for the core domains and the Lebanon corpora, and the correctness of those assertions.

The data in Table 5.2 represents all the assertions produced by REDMAP aggregated for all texts in a given corpus. However, some interpretations share content. For example, if a text discusses the candidates running in an election, and a subsequent text discusses the voter turn out and the winner, the two interpretations are likely to share assertions about the election, and the winner being a candidate. Looking at only the unique assertions produced by REDMAP, eliminating duplicate assertions across texts, removes 7% of the non-\textit{isa} assertions from the core domains corpus, and 3% of the non-\textit{isa} assertions in the Lebanon data. A large number of \textit{isa} assertions were repeated across stories, 20% in the core domains, and 25% in the Lebanon texts. The Lebanon texts shared more ontological assertions (such as \textit{isa}) because they shared more actors and locations by design, although they shared relatively little non structural content. Factoring out the approximately 15% of assertions that were duplicated across stories has no significant effect on the ratio of relevant or irrelevant assertions produced; the distributions stayed effectively the same for both corpora.
5.2.2 Interpretation Coherence

Measuring the accuracy of the assertions produced only tells part of the story regarding the quality of an interpretation. Being able to build representations for people, places, and events that span multiple sentences and multiple texts requires building coherent representations. Since all corpus paragraphs were about one primary topic (person, place, event, etc.), or a connected set of events (for stories that discussed more than one event), it is assumed all REDMAP interpretations should result in an interconnected, coherent, set of assertions. Two assertions are deemed interconnected if they share at least one argument. To measure coherence, each interpretation was clustered into groups of assertions. An assertion was placed in a group if it shared an argument with any other assertion in the group.

For example, the assertions in the interpretation fragments F1, F2, and F3 would be clustered in the following manner. All of the assertions in F1 would be placed in one assertion cluster. The first assertion in F1 shares a term with the second assertion in F1, and the second shares a term with the third, fourth, and fifth. The assertions in F2 would also be added to this cluster, because the second assertion shares a term, Israel, with assertions from F1 already in the cluster. The first assertion in F2 is then included because it shared a term with the second assertion in F2. The assertions in F3 would form their own second assertion cluster, because none of them share terms with any of the assertions in the other cluster, however all the assertions in F3 share terms with each other. With only what is depicted these three fragments would form two assertion clusters. However if more representation from the understanding of T1 was included, for
example, that the speaking was about the leaders then it is possible all assertions would collapse into one cluster.

For evaluation it is assumed all texts represent interconnected content. An optimal and correct interpretation would have exactly one assertion cluster. There may be cases where a text is properly represented with more than one assertion cluster, however for the purposes of this analysis it is assumed a correct interpretation should have only one cluster. There are two reasons for multiple clusters to be formed incorrectly. The first is due to the absence of the connecting information. This could be because a sentence is not matched or because only partial understandings were produced which left out critical pieces. For example, with just what is depicted by F1, F2, and F3 of the interpretation of T1, two clusters are formed, as there is no connection between the leaders in the bunker, and the bombing or the speaking.

The second source of failure is due to either incorrect structures being produced, or a failure in the reference resolution algorithm in forming a reference mapping. For example, if in the interpretation of T1 a connection was not made between the dropping of the bombs in the first sentence and the attack in the second sentence. The resulting representation would then be fragmented. Similarly if a component of the interpretation was misrepresented this might block the reference resolution algorithm from forming a connection, again resulting in multiple fragments.
The number of errors can be measured by counting the number of assertion clusters per text minus one, as there should always be one assertion cluster. An interpretation producing zero assertions is reported as producing zero assertion clusters as well.

For a third of the texts evaluated, REDMAP produced a unified interpretation with only one assertion cluster. For another third of the input, REDMAP missed one piece of connecting
information resulting in the interpretation being cut into two pieces. The aggregate data for all stories can be seen in Figure 5.1. While intuitively the longer a story is the more opportunities
REDMAP has to make a mistake, there is only a weak correlation between story length in sentences, and the number of assertion clusters (Kendall’s tau 0.34, p value < 0.001).

Figure 5.2, shows the breakdown of number of assertions clusters by corpus section. For example, 63% of the geography stories are processed without a single error, while the remaining 37% had only one error. Given that the geography stories are about 5 times as long as the other stories in the corpus they have more opportunity for error, however still perform well. Clearly topic has an affect on coherence, as some domains are understood better than others. In contrast to the relatively good performance of the geography stories, the terrorism stories most frequently produced two assertion clusters (44%), and had one text produce six clusters, and another seven.

### 5.3 Partial Understandings and Interpretation Quality

Some domains have more linguistic information in ResearchCyc than others. Since REDMAP will produce partial understandings in less understood domains it is important to evaluate the effects of weak linguistic coverage on quality. An experiment was run to evaluate the relationship between the amount of text understood (partial understanding) and the quality and quantity of assertions produced. This provides evidence for claim 4:

**Claim 4:** In domains that have fewer or less specific patterns REDMAP can piece together partial understandings at a cost to the accuracy and quantity of assertions produced. This relationship can be predicted to some degree from the amount of text understood.

For REDMAP to understand a piece of text two things must happen. First the words in the text must be identified and mapped to lexical concepts, semantic concepts, or named entities (proper
nouns). Then those concepts must be matched by one of REDMAP’s rules to produce an interpretation fragment. It is also possible for rules to match literal stings, in which case REDMAP would be able to understand a word without it occurring in its lexicon. These processes are discussed in detail in chapter 3.

5.3.1 Lexicon and Rule Coverage of Corpus

The lexical coverage of the words in each section of the corpus is presented in figure 5.3. For 100% of the military, election, and geography subsets REDMAP was able to identify at least 90% of the words in the texts. The text with the fewest identifiable words still identified 68%. The mean over all of the texts is 97% and the median 100%. While patterns could match strings explicitly and bypass the need for identification via the lexicon, this for the most part represents an upper bound on the amount words which can be picked up by patterns. This data also shows that the lexicon is not a limiting factor for REDMAP to form understandings.

Even though a word occurs in REDMAP’s lexicon, REDMAP may not be able to use it. For a word to contribute to REDMAP’s interpretation of the text it must then be matched by a rule that successfully completes and produces an interpretation fragment. Figure 5.4 shows histograms of the percent of words in a text contributing to pattern matching rules that contributed to the final interpretation of a text. This graph can be read in the same way as figure 5.3.
Although most of words in the corpus occur in REDMAP’s lexicon, the ability to understand these words, via pattern matching rules, varies widely by corpus section. The terrorism and election texts, for example, used no fewer than 56% and 60% of the words in a text. The median ratio of words used was 78% in the terrorism subsection and 90% for the election texts. On the
other end of the spectrum, the understandings of the relation and organization texts used at most 70% and 80% respectively of the words in a text. Overall the mean and median value was 59% of words used in understanding a text. The inner quartile range (50% of the texts) was
understood using from 44% to 77% of the words in the text. Clearly there is a wide range of partial understanding for the texts in the corpus.

### 5.3.2 Effects of Partial Understanding

An evaluation was conducted to quantify the effects of varying levels of partial understanding on the quality and quantity of assertions in REDMAP interpretations. The evaluation measured whether it was better for REDMAP to give up when few words were understood in a sentence, or try to construct a partial understanding.

This was done by setting a threshold for the number of words used in pattern matching rules for a sentence. For example if the threshold was set at 80%, and 90% of the words in a given sentence were used in pattern matching rules, the results of those rules were integrated in the running interpretation of the text. However, if instead only 75% of the words in the sentence were used in pattern matching rules their output was discarded, the sentence was skipped, and REDMAP moved on to the next sentence in the text. For discussion this threshold will be referred to as the *words-used threshold*.

The words-used threshold was systematically varied and 5 sections of the extended corpus were evaluated, including text that introduced new named entities (refer to section 2.2.3). The results show the affects of partial understanding on the quality of interpretations. Figure 5.5 shows the trade off between quality of assertions when REDMAP is prevented from using smaller partial
Figure 5.5 Quality of the assertions being produced by REDMAP vs. partial understanding threshold.

Figure 5.6 shows how the number of assertions produced by REDMAP is also heavily affected by the ability to leverage partial understandings.
5.3.3 Discussion

For the domains in which REDMAP has better text coverage (e.g. elections and terrorism), it has patterns that cover most of the input and therefore always produces decent results regardless of threshold. For other topics where REDMAP falls back on partial understandings more often,
such as history and Lebanon, the amount of text understood has a clear relationship to quality and quantity of assertions.

It should be noted that flat horizontal tails running to the left indicate the point at which change ceases to occur. For example in the election data every point to the left of 0.3 is effectively identical. Even when the threshold is set at 0.2, it is still using at least 30% of the words in most of the input sentence.

One reason processing becomes slightly worse when the words-used threshold is set above 0.7 is that some of the patterns, such as those that introduce new names, function accurately even though they are skipping a word or two for most of the sentences they match in the corpus. When the words-used threshold is set to the point where skipping a word or two will result in the sentence being dropped, then these interpretation fragments are not produced. In the case of failing to introduce new names this effect cascades through the rest of the corpus section, as the entity will not be recognized in subsequent text. This is especially true of election stories which by their nature involve numerous names, such as those of all the candidates.

There is a clear relationship between the quality and quantity of assertions produced and the number of words used during pattern matching. This is evidence in support of claim 4: that REDMAP can predict the quality of its output. However, because there is a tradeoff between quality and quantity when partial understandings are used, it is not clear what setting for the words-used threshold is best. Where to operate depends on which type of error is more
appropriate for the language understanding task being targeted. A task that desires more accurate assertions should raise the threshold, while a task that desires quantity and is more noise tolerant should lower it.

5.4 Question Answering

While the previously discussed evaluations measured the relative correctness of the assertions being produced by REDMAP, they did not directly measure the usefulness of those assertions for any specific task. A parameterized question system (Forbus et al. 2007; Cohen et al. 1998) was used to generate “who?”, “what?”, “where?”, and “when?” questions, as appropriate, for all correctly identified entities mentioned in REDMAP interpretations of texts from the corpus. Only the correct entities were used because it makes no sense to evaluate question answering performance on gibberish entities.

A back-chaining system was used to attempt to answer the automatically generated questions. It used 787 axioms based on the generalization hierarchy of predicates in the ResearchCyc ontology. The answers produced by this system were marked as correct or incorrect. The same questions were asked of the knowledge base both before and after reading the input stories. This was done to differentiate between knowledge already known and that which was added by REDMAP.

Table 5.3 shows the results of the question answering evaluation. This evaluation was not broken down by subsection of the core domains corpus, the Lebanon corpus was produced and
evaluated at a separate time. These results show that REDMAP-generated assertions were able
to contribute significantly to question answering. For the entities mentioned in the corpus,
REDMAP-generated knowledge allowed for three and one half times as many questions to be
answered as could be done with the original knowledge base alone. Over 93% of the answers
provided by REDMAP knowledge were correct.

In addition to evaluating the knowledge explicitly produced by REDMAP, these assertions were
also used as the starting point for a deductive reasoning system, another component of the
Learning Reader. The quality of the assertions produced by this combined system was also
evaluated using the question answering system. (Forbus et al. 2007) These results are also
included in Table 5.3 under the condition Reading + Deductive Reasoning. The deductive
reasoning system used a set of 1,978 axioms, an order of magnitude larger than those used in
question answering. These axioms included inferences more complicated than just moving up or

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Condition</th>
<th>Questions Generated</th>
<th>Questions Answered</th>
<th>% Questions Answered</th>
<th>Incorrect Answers</th>
<th>Accuracy of New Questions Answered</th>
<th>Cumulative Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Domains Corpus</td>
<td>Before Reading</td>
<td></td>
<td>109</td>
<td>10.7%</td>
<td>0</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Reading</td>
<td></td>
<td>381</td>
<td>37.4%</td>
<td>18</td>
<td>93.4%</td>
<td>95.3%</td>
</tr>
<tr>
<td></td>
<td>Reading + Deductive Reasoning</td>
<td></td>
<td>455</td>
<td>44.7%</td>
<td>18</td>
<td>100.0%</td>
<td>96.0%</td>
</tr>
<tr>
<td>Lebanon Corpus</td>
<td>Before Reading</td>
<td></td>
<td>47</td>
<td>8.0%</td>
<td>0</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Reading</td>
<td></td>
<td>181</td>
<td>30.8%</td>
<td>9</td>
<td>93.3%</td>
<td>95.0%</td>
</tr>
<tr>
<td></td>
<td>Reading + Deductive Reasoning</td>
<td></td>
<td>202</td>
<td>34.4%</td>
<td>9</td>
<td>100.0%</td>
<td>95.5%</td>
</tr>
</tbody>
</table>

Table 5.3 Results of question answering evaluation for core domains corpus and the Lebanon corpus.
down the generalization hierarchy, and the deductive chains were permitted to be twice as long as those in the question answering system. While this deductive reasoning system is not part of this work, it demonstrates that errors introduced by REDMAP do not compound when channeled through additional deductive reasoning and that additional useful knowledge can be produced from REDMAP-generated knowledge. This shows that REDMAP-generated knowledge can be used by multiple different reasoning systems.

5.5 Discussion

Contrasting the assertion accuracy results and the question answering results shows that REDMAP-generated assertions are more accurate for question answering than they are overall. This seems paradoxical. The interpretation coherence results provide a clue: many of the interpretations produced are fragmented. These results imply that most of the errors encountered in the assertion accuracy results are on assertions that do not touch known entities. These erroneous fragments are connected in the knowledge base primarily to collections in the ontology, but not known entities. In such a state, they have no affect on processes that do not go looking for them. For example, question answering asks only about the details of entities mentioned in the stories. Doing so will not encounter one of these fragments. If however, the knowledge base was probed with other kinds of questions, like “how many elections do you know about?” or “list all the attacks that occurred in Iraq?” These fragments may then be uncovered, and lead to erroneous answers. Varying the word-used threshold causes a tradeoff to occur between quantity and quality of assertions produced. It is possible this could also be used
to control the number and size of these erroneous fragments in the knowledge base. Further experimentation and evaluation would be needed to verify these hypotheses.
Chapter 6  Scalability: Ambiguity and Speed

Understanding language requires managing and resolving ambiguity, and the methods used to do so can have a large impact on the scalability of an approach. REDMAP can defer the resolution of ambiguity, and since this approach also introduces episodic ambiguity to the set of options being tracked, the amount of ambiguity could grow large. REDMAP queries to memory to resolve ambiguity, and therefore the scalability of an approach that is constantly interacting with a knowledge base is an appropriate concern. The following four experiments evaluate the scalability of incremental language understanding with respect to a large underlying memory (knowledge base) over a corpus of texts.

The first evaluation contrasts the scalability of three models for performing incremental understanding. The second experiment evaluates influences on REDMAP’s rate of reading. This experiment provides support for the second claim made in this dissertation.

Claim 2: The reading rate (mean time per sentence for a text) of the REDMAP algorithm is empirically independent the number of references in the text and the length of input (number of sentences).

The third experiment evaluates the effects of existing episodic memory on the amount of ambiguity evaluated by REDMAP and the time taken to do so. An unexpected result of this evaluation was support for the lesser third claim made in this dissertation.

Claim 3: Complexity of syntax is not a predictor of processing time in REDMAP.
The final evaluation is an expansion of the third. It contrasts REDMAP’s performance when it is reading about something in episodic memory, versus when it is not. The results of this experiment provide strong support for the first claim made by this dissertation.

Claim 1: A memory-based reference resolution algorithm (REDMAP) can provide broad coverage of and extending an existing large knowledge base by grounding to existing episodic memory as it parses and can use that memory to reduce ambiguity.

6.1 Incremental Updates

A crucial factor in the time taken to process text is the level at which ambiguity is addressed and tracked. Since REDMAP performs incremental updates of its understanding, the size of the increment roughly corresponds to the size and scope of the ambiguity being deferred or resolved. In a non-exhaustive search for understanding, this decision also affects the points at which backtracking can occur and the amount of information available to heuristics to make the decision. The goal was to produce an implementation that forced as few requirements on DMAP as possible as to the amount of ambiguity that could be tracked and when it must be resolved. Allowing decisions to be deferred longer could allow for more information to be brought to bear in resolving ambiguity. Three variations of the REDMAP algorithm were produced, which make different decisions regarding how to track and resolve ambiguity. This section continues with a description of each approach, and then ends with an experiment contrasting their processing speeds.
6.1.1 Word Level Exploration (WLE)

The first REDMAP algorithm exhaustively explored ambiguity at the word level. In word level exploration (WLE), running interpretations were updated after a semantic fragment of any size was identified by the mappings in the lexicon or pattern matching rules. All possible running interpretations that could be produced were produced and tracked. For example, consider the phrase “marine forces”. In ResearchCyc “marine” has two denotations (MarinePersonnel and Sea) and “forces” has four, two as a noun ((GroupFn Agent-Generic) and ModernMilitaryUnit-Deployable) and two as a verb (ViolentAction and CoercingAnAgent). In WLE a state was created for each denotation. Hence with the input “marine forces”, “marine” would generate 2 states one for each denotation, and “forces” would split each of those states 4 ways, resulting in 8 separate states being tracked.

6.1.2 Sentence Level Exploration (SLE)

Language at the level of words is highly ambiguous. For example, the word “bomb” could be representing, among other meanings, an explosive device, or an event as in “to bomb” it is impossible to tell from the word alone which one is meant. However, in the context of a sentence or phrase its meaning becomes at least less ambiguous, if not unambiguous. Consider the following sentence.

The soldiers drove tanks into Lebanon.
Although the word “tank” could be interpreted to mean LiquidStorageTank, it is meaningless to “drive” such an object. No REDMAP patterns can produce such an interpretation, therefore creating a new state to represent this sense of the word “tank” is not productive. The second algorithm takes advantage of this and does not generate a new running interpretation for each word unless there are phrasal patterns where the distinction is relevant. However, it does generate a running interpretation for every possible sentence level ambiguity. This algorithm is named sentence level exploration (SLE).

### 6.1.3 Reuse Driven Exploration (RDE)

SLE exhaustively tracks ambiguities that arise at the sentence level. A third algorithm was developed, building on SLE, to attempt to selectively direct the search at the sentence level. This algorithm uses best-first search biased to prefer states that have a greater percentage of their assertions known to the knowledge base and are constructed from fewer assertion clusters (described in the previous chapter). Put another way, preference is given to states with the least new information and that have more references merged with each other and with memory. This algorithm is named reuse driven exploration (RDE). The system and algorithms described in chapters 3 and 4 are the latest version of RDE. Ideally RDE will proceed more directly to the correct interpretation, and do so without getting caught in local maxima/minima representing suboptimal or incorrect interpretations.
6.1.4 Scalability of Incremental Updates

An experiment was conducted to measure the time to process the corpus (from input text through producing an interpretation) for each of the three ambiguity tracking algorithms: WLE, SLE, and RDE. This experiment was performed on the core domains corpus described in chapter 2, before the Lebanon corpus was added.

The data shown in Figure 6.1 plots the cumulative distribution of the number of sentences that were processed under a given time for each of the algorithms. A faster algorithm will appear closer to the upper-left corner. A point at 60% on the y-axis indicates that 60% of the sentences in the corpus were processed at least as fast as the corresponding time on the x-axis. For example, for RDE 60% of sentences were processed in under 574 ms each, while 5/6 of those (50% of the sentences overall) were processed in under 315 ms each. Only 10% of the sentences for RDE took between 315 and 574 ms. Time, depicted on the x-axis in milliseconds, is shown in log scale.

This evaluation shows that WLE asymptotically approaches being able to process only 63% of the corpus, even when given several hours to read a sentence. This implementation of WLE is impractical. The fact that the data reaches a plateau illustrates the exponential nature of this algorithm and does not speak well of WLE in general. SLE fared better, and was able to process 90% of the sentences in the corpus in under 16 seconds each, and 99% when given up to 8.7 minutes per sentence. While this rate of reading may be usable for an offline reading algorithm, its performance is still far too slow for an interactive reader. RDE performed the best and was able to process 93% of the corpus in under 4 seconds per sentence, and 99% with a little over 2
minutes per sentence. Comparing the distributions with a Wilcoxon rank sum test indicates that RDE represents a significant improvement over SLE ($p < 1e-06$). The difference between the median times for the two distributions is 90ms. Based on these results WLE and SLE were deemed not worth continued investigation. All future improvement and evaluation was done
using only an RDE-based implementation. The system described in chapters 3 and 4, and evaluated in chapter 5 and the remainder of this chapter, is the latest version of RDE.

### 6.2 Influences on Rate of Reading

Several factors can affect REDMAP’s processing time. Pattern matching time is linear in the number of rules matched, and the longer a sentence is the more rules which could potentially match. Different subsections of the corpus have different levels of coverage and partial understandings produced by different sets of rules matching (see chapter 5), and these sets of rules and the interpretation fragments they produce could behave differently. The reference resolution algorithm described in chapter 4 is exponential in the number of sentences read over the average number of interpretation fragments produced by pattern matching. The running time of this algorithm is further compounded by the process of selecting a reference mapping, which is exponential in the number of references. REDMAP therefore has the possibility of slowing down as more patterns are matched, more references are introduced, or more sentences are read.

This raises the following questions regarding what factors influence REDMAP’s running time.

1. Is interacting with a large knowledge base is tractable?
2. Is REDMAP tractable for the multiple subsets of the corpus?
3. How does the number of rules matched affect processing time and rate?
4. How does text length affect the number of rules matched and processing rate?
5. How does the number of reference affect reading rate?
An experiment was conducted by processing the extended corpus (described in chapter 2) and recording the reading rate for each text. Since there is some variation in reading time, each text was read three times and the median time is used. No state is preserved between reading the story each time; each reading of a text is an independent trial. The median is used in order to be robust to outliers, which sometimes occurred when a query into the knowledge base would take abnormally long to return. The median time is also consistent with the min time, and as more samples are collected the mean approaches the median/min time. Only these median times are reported here.

6.2.1 Overall Reading Rate

The rate of reading each text in the corpus was computed. Since texts varied in length by number of sentences (median 3, max 21), time to process a text was normalized by the number of sentences it contained. Normalization was done by dividing the time to read a text by the number of sentences in the text, computing the average time per sentence for the text. The median reading rate for all texts was 300ms/sentence, while the mean was 1,051ms/sentence. The data follows a Zipf-like distribution, and figure 6.2 shows a plot of the cumulative distribution with average reading rate per sentence on the x-axis, and the percent of texts in the corpus read at or under that rate on the y-axis. 75% of texts are read at a rate under one second per sentence, and 90% at or under a rate of 2.8 seconds per sentence. These rates are reasonable for real-time reading goals.

Figure 6.2 aggregates the data for all nine subsections together. Looking at each corpus section independently shows only four that deviate significantly, p values less than 0.05, when compared
Figure 6.2 Mean reading rate per sentence in milliseconds on the x-axis, vs. the percent of texts read at or under that rate on the y-axis. Vertical lines at 50%, 90%, 95%, and 99%. The x-axis is clipped at 10 seconds; there are two points missing, 11,636, and 18,773.

with a Wilcoxon rank sum test to the overall distribution. The election and history stories were processed faster than the overall distribution, and are skewed further toward the upper left corner. In contrast the Lebanon and geography stories follow slightly slower curves when compared to the aggregate data. Overall there was a wide range in measurements with the Wilcoxon test. However, on inspection it can be seen that behavior is consistent across corpus
sections. There is some slight variance from the position and slope of the cumulative distribution curve depending on the domain, but all domains are being processed at similarly. The cumulative distribution curves for each section of the corpus are depicted in Figure 6.3.
While there is some variation across domains, the distributions are very similar to each other. The election stories are very well covered by REDMAP rules and they perform the fastest. As shown in the partial understanding experiment in the previous chapter, REDMAP has a wide range of patterns to extract partial understanding from the history domain and this increased flexibility likely explains the difference in performance. The Lebanon stories are the densest in the corpus, having the most words and references per sentence, representing content that is more semantically rich and complicated. For example, Lebanon stories discuss multiple detailed events per text as well as how those events are connected. In contrast, a terrorism story only discusses one particular terrorist attack, necessitating far less reference and ambiguity resolution. Therefore it seems consistent that the Lebanon stories would appear on the slower end of the spectrum. Overall REDMAP appears to be tractable and behave similarly regardless of corpus section.

6.2.2 Pattern Matching Time

The number of REDMAP rules matched correlates well to the time spent matching rules. The Pearson’s correlation coefficient for the relationship is 0.84 (p value < 0.001). It is not a perfect correlation because time spent partially matching a rule that does not complete is also charged here. The mean time to match a rule is 4.7ms, and the median is 3.5. This time also accounts for the majority of time REDMAP spends processing text. The Persons correlation coefficient between the total processing time and the number of matched rules is 0.82 (p value < 0.001). This data shows that REDMAP’s processing time is heavily influenced by the number of rules
matched. Rule matching is discussed in chapter 3, and there is ample room for improving this process to reduce running time.

The current implementation produces a mean of 344 and a median of 77 rule matches per sentence averaged over all texts in the extended corpus. These rules are matched from a pool of 184,007 rules. REDMAP appears to function well with large sets of rules, however no specific studies have been designed to evaluate effects of the quantity of rules REDMAP is provided.

Interpretation fragments produced by matched rules have an exponentially combinatoric impact, in the worst case, on the rest of REDMAP. However, the time being spent here is on the actual matching and not the effect of the fragments produced. With the exception of introducing a new name, there is no relationship between pattern matching in one sentence and any state produced by sentences that were processed before it. It is therefore not surprising that there is no relationship between the time to process rules, and the number of sentences in a text. Pattern matching time is a property of the sentence being processed.

### 6.2.3 Effect of Text Length

In the worst case, REDMAP processing time is the average number of interpretation fragments, which maps directly to the number of rules that matched, raised to an exponent corresponding to the number of sentences. If there was a systematic slowdown in processing time, this would predict a relationship between the rate of reading for a text and the number of sentences in that text. No such relationship is observed in the data from reading the corpus. The Pearson’s
correlation coefficient between the number of sentences in a text and the mean reading rate per sentence in the text is 0.14 (p=0.01), which shows no linear correlation. Kendall’s tau, a non-parametric test for correspondence, on the same data is 0.12 (p=0.004), also indicating no correlation. Reading rate does not appear to be influenced by text length. The possibility of ambiguity compounding across sentences, and thus slowing down the rate of reading, is not observed in this algorithm operating on the given input corpus.

6.2.4 Effect of References

The reference resolution algorithm is exponential in the number of references which need to be resolved. If REDMAP does not scale with the number of references this would predict a relationship between processing time and any of the following three factors: number of reference groups in an interpretation, number of interpretations evaluated, or number of reference mappings evaluated. Computing Kendall’s tau for these three relationships yields the following weak, although significant correspondences (p values < 0.001): reference groups, 0.29; interpretations, 0.37; and reference mappings, 0.37. While there is weak evidence of a relationship, it offers no discernable predictive power. These factors conform to no shapes, much less a curve, when plotted against reading rate. REDMAP scales independently of factors relating to the number of references.
6.2.5 Discussion

The data shows that for the corpus evaluated, REDMAP is capable of operating tractably over a large knowledge base. Although processing rates vary slightly for subsections of the corpus, REDMAP behaves consistently and tractably for all nine sections. The predominate factor in REDMAP processing time relates to the number of rules matched. Rule matching is influenced only by the contents of a sentence, and therefore scales independently of other text-level properties, like number of references or number of sentences. The methods used by REDMAP to interact with memory are not a limiting factor in REDMAP’s scalability. The results of this evaluation directly address the second major claim made in this dissertation.

Claim 2: The reading rate (mean time per sentence for a text) of the REDMAP algorithm is empirically independent the number of references in the text and the length of input (number of sentences).

6.3 Effects of Existing Knowledge

The goal of using a memory-based, reuse-driven algorithm for language understanding was that it should be able to leverage what it knows in order to read better. A small experiment was designed to evaluate whether REDMAP could leverage existing episodic knowledge in order to read more efficiently both in terms of metrics of ambiguity and processing time. (Livingston and Riesbeck 2009)
6.3.1 Input

A simple story about an Al-Qaida attack in Afghanistan was rewritten as three different paraphrases.

**Story A**
An attack occurred in Afghanistan.
The bombing was performed by Al-Qaeda.
The attack occurred on July 18, 2008.
The attack targeted United States soldiers.

**Story B**
There was an attack on July 18, 2008.
The bombing occurred in Afghanistan.
Al-Qaeda targeted United States soldiers.

**Story C**
Al-Qaeda performed an attack on July 18, 2008.
United States soldiers were targeted by the bombing in Afghanistan.

Each text represents the same event and presents the same amount of detail. All three can be understood by REDMAP as the following representation in ResearchCyc.

```
(isa Bombing-1 Bombing)
(eventOccursAt Bombing-1 Afghanistan)
(dateOfEvent Bombing-1
  (DayFn 18 (MonthFn July (YearFn 2008)))))
(performedBy Bombing-1 Al-Qaeda)
(intendedAttackTargets Bombing-1 Group-2)
(isa Group-2 (GroupFn MilitaryPerson))
(isa Group-2 (GroupFn (MemberFn (ArmedForcesFn UnitedStatesOfAmerica)))))
```

Although there is an identical set of predicate logic assertions passing through the reference resolution algorithm, there are fewer references to resolve in Story C, then there are in Story B, and fewer in Story B then there are in Story A. For example, the second sentence of Story C will
produce two sets of assertions, one representing a bombing that targeted US soldiers, and another representing a bombing that occurred in Afghanistan. Due to the way the pattern matching occurred, the bombing in both sets of assertions is already represented as the same event, resulting in effectively one event reference for the whole sentence. For all stories, for each sentence, REDMAP must merge the event reference in that sentence to the reference group representing the event in the running interpretation.

6.3.2 Experiment

An experiment was conducted by reading each of the stories with and without this representation already present in episodic memory. When the knowledge exists in episodic memory REDMAP should be able to retrieve the same representation for all three variations in input form. The hypothesis is that REDMAP should be able to leverage existing knowledge to evaluate fewer parsing states and reference mappings, and do so in less time than when it sees a story without existing episodic knowledge.

Three different trials were run. In the first trial story A was read and the representation it produced was stored in memory, then story B and C were read with access to the assertions produced from reading story A. This was also done reading story B first, storing the result, and then reading C and A. Similarly, C was read first, than A and B.
6.3.3 Results

Not surprisingly REDMAP processed each story identically when the episode was already in memory regardless of which story produced the existing knowledge for it to leverage. The data presented here aggregates all of these runs. Only one run for each story is used in each of the conditions: reading for the first time, and reading with the episode already in memory. The results are provided in table 6.1, all values are normalized by the number of sentences in the input text. The columns labeled first time reading represent REDMAP’s performance when encountering that particular text without the representation already existing in episodic memory. The columns to the immediate right of those columns represent the difference when episodic knowledge is available. A negative difference means that REDMAP’s performance was improved by using episodic knowledge. Two times are shown, the time spent only in the reference resolution algorithm, and the total time for all processing, including reference resolution.

Two values are provided as measurements of ambiguity. The first is the number of parsing states or interpretations created by REDMAP. The second is the number of reference mappings evaluated by REDMAP when attempting to merge the assertions from a sentence with the running interpretation. While it might seem inconsistent that REDMAP would have evaluated fewer reference mappings per sentence than parsing states, this is explained by the fact that there are no mappings to be evaluated for the first sentence, since it is being merged with an empty representation.
The hypothesis was that the availability of existing episodic knowledge would allow REDMAP to evaluate fewer parsing states and reference mappings, and do so faster than it could without the episode in memory. Figure 6.4 shows that REDMAP was able to reduce the amount of work necessary to resolve references in every measure, except the number of parsing states for story A.
Figure 6.5 Relative reference resolution and total time per sentence for each story when the relevant episode already exists in memory. Note the scales are not the same for both graphs. Horizontal line at 1.0 indicates the value of no change. The graph for reference resolution time also has a horizontal line at 2.0, indicated a double value in time.

While REDMAP was able to leverage episodic knowledge in order to resolve references more directly, the added time to interact with this memory mitigated any benefits gained for two of the three cases. Figure 6.5 shows the relative amount of reference resolution and total time spend to understand each story with the episode already in memory. Stories A and C were processed slower, while overall story B was processed faster even though reference resolution took longer.

6.3.4 Discussion

Reference resolution time is fairly fast in all cases before ground instances exist in the knowledge base. In processing Story A, REDMAP evaluates more ambiguity than B or C, and this corresponds to increased run time as well. Although Stories B and C evaluate the same
number of semantic ambiguities, story C spends more time figuring out how to resolve the references between sentences.

In all three stories (although less so in Story A), REDMAP was able to leverage existing episodic memory in order to reduce the number of ambiguities explored. This is a positive result. When REDMAP had references in memory to aid its decision making it used 44% fewer states for story B, and 22% less for story C. These stories have more ambiguity packed into individual sentences and therefore pruning opportunities are more available and more beneficial than for story A. The reduction in reference mappings is even more significant. For stories A and C there was a 15% and 20% reduction respectively. REDMAP was able to eliminate 74% of the reference mappings that needed to be evaluated for Story B when it was able to leverage existing episodic knowledge.

However, in contrast to the positive result of reducing the amount of ambiguity evaluated, when references are available in memory, REDMAP spends more time exploring each ambiguity, nearly twice as long. The slowdown is somewhat mitigated by a more direct path though the search space, although REDMAP still operated at a net loss across all three texts with respect to run time. Overall, while not being entirely positive, this result is promising, and warrants further investigation, which is provided by the last experiment presented in this chapter.

The large range in processing times per sentence between the different texts can be explained by variance in the quantity of output produced by pattern matching. Longer sentences, such as
those in Story C, have the opportunity to produce far more interpretation fragments, and thus an increase in parsing time. Story A produces approximately 300 interpretation fragments, Story B only 150, and story C over 650 interpretation fragments. The reference resolution algorithm described in chapter 4, is able to rapidly process these fragments, or prune them, as can be seen by the small processing times for reference resolution, regardless of text.

Even though Story B has more sentences than Story C (but not more words), and is syntactically more complicated than Story A, REDMAP was able to read it by processing the least number of parsing states and reference mappings, and at the fastest rate. This is counter to the assumption made to justify the use of simplified English for the Learning Reader project. This provides weak support for the third claim of this dissertation.

Claim 3: Complexity of syntax is not a predictor of processing time in REDMAP.

Further and far more detailed investigation of this claim is warranted.

6.4 Impact of Existing Knowledge Structures

REDMAP encounters ambiguity in two primary places. The first is selecting an interpretation fragment from the pattern matching to merge with the running interpretation to create a new parsing state. The second is determining how the references in the new interpretation fragment are mapped to the existing references. If REDMAP is able to leverage existing knowledge structures to read more efficiently, this would predict the exploration of fewer states, and the
evaluation of fewer reference mappings. The experiment described in section 6.3 indicated that REDMAP could leverage existing episodic knowledge to reduce the amount of ambiguity it needed to evaluate. However it was left unclear whether time could be saved in doing so, due to the increased cost of interacting with memory. This experiment was designed to evaluate the affect of leveraging existing episodic knowledge across all the stories in the corpus.

To evaluate the relative overhead or savings attributable to REDMAP for leveraging memory, each text in the corpus was read under two conditions. After the story was read the first time its interpretation was stored in memory, and then the story was read again. Thus the differences between having and not having access to existing knowledge structures can be evaluated. Ideally REDMAP should be able to recognize and understand the story with less work (exploring fewer parsing alternatives, and evaluating fewer reference mappings) in less time when it has existing knowledge structures in memory.

This experiment has two conditions, seeing the story for the first time, and seeing the story after assertions representing its content are stored in the knowledge base. As in the previous experiment, each text was read three times in each condition and the median time is used. No intermediate parsing state is preserved between reading the story each time. The only difference is between conditions, when the assertions representing the content of the story are added to the knowledge base so that they are potentially available to REDMAP when re-reading. (See the discussion of the RDE algorithm earlier in this chapter, and the process of reference resolution discussed in chapter 4 for the details of how REDMAP interacts with existing knowledge.)
In addition to recording the processing time for each text, the number of parsing states explored and reference mappings evaluated were also recorded to be compared between the conditions. The timing data allows comparison between conditions for speed. The other data allows comparison between conditions to see if REDMAP could more directly recognize and understand the text when reading it with the episode in memory, by leveraging it to eliminating or avoiding potential ambiguities.

6.4.1 Existing Knowledge versus Ambiguity

Of the 295 texts in the corpus, when knowledge structures were available in memory, REDMAP explored a different number of states or evaluated a different number of reference mappings for 125. Table 6.2 presents the difference in amount of ambiguity explored for those 125, when REDMAP had access to existing knowledge structures contrasted to when it did not. Since texts vary in number of sentences this data is normalized by the number of sentences in a text.

The data shows that by leveraging existing knowledge structures REDMAP was able to on average explore fewer parsing states and reference mappings per sentence. The columns in Table 6.2 label “first time reading” represent the number of states or reference mappings explored per sentence when knowledge structures were not available in memory to use. The adjacent columns labeled “difference with existing knowledge” represents the changes in value for reading in the second condition when existing knowledge is available. A negative change means that REDMAP was able to operate more efficiently by evaluating that many fewer states or mappings. A positive value would mean that REDMAP did more work.
Table 6.2 Amount of ambiguity when reading extended corpus with and without existing episodic knowledge for 125 texts that differed between conditions.

While on average REDMAP was able to explore fewer parsing states, comparing the distribution of parsing states between conditions with a Wilcoxon rank sum test does not show a significant shift (p value 0.6). The algorithm for selecting parsing states (described in chapter 4) does not behave significantly different between the two reading conditions evaluated in this experiment.

For the 125 texts REDMAP significantly reduces the number of reference mappings that need to be evaluated when existing knowledge structures are available (Wilcoxon rank sum test p value 0.03). The mean difference was -7.1 and the median -0.7. Figure 6.6 shows the range of difference for each section of the corpus. These results are computed considering only the 125 texts. For those that differed, the difference was significant. If the distributions for all 295 texts, including the 170 texts that did not change, are compared the mean difference is lowered to -3 reference mappings per sentence, and the median, is of course 0. The significance computed by
Figure 6.6 Boxplots of the relative number of reference mappings evaluated per sentence for each story when the relevant episode already exists in memory. Horizontal line at 1.0 indicates the value of no change. Values below the line indicate a relative savings. Thick bars in the center of the boxplots represent median values. This graph includes only the 125 texts that changed.

A Wilcoxon rank sum test is also decreased (p value 0.23), although this still suggests a potential shift in the distribution, even when heavily biasing it with 170 texts that did not change.

The hypothesis was that REDMAP could leverage existing knowledge in order to explore less ambiguity. In terms of reference mappings, for those that changed, REDMAP explored significantly fewer mappings when existing knowledge was available. The metrics for parsing states per sentence showed a reduction for ambiguity in terms of absolute numbers, however the shift is not significant for all texts (median = 0, mean = -3.1, Wilcoxon rank sum test p-value = 0.72).
6.4.2 Existing Knowledge versus Time

The second part of the hypothesis was that REDMAP could perform reference resolution in less time when it has existing episodic knowledge available. This hypothesis is not borne out in the data. Overall there is a slight run-time cost attributable to DAMP leveraging existing knowledge structures. For all 295 texts there was a mean 27.7 ms/sentence, median 5ms/sentence, slowdown in reference resolution when a story was read for a second time. Comparing the two conditions with a Wilcoxon rank sum test shows this difference, although small, is significant (p value < 0.001). Looking at the time for all processing done by REDMAP there was a mean -5.4 ms/sentence acceleration when reading for the second time, although the median value was a 3.3 ms/sentence slow down. The difference in total time data does not represent a significant shift in distributions when compared with a Wilcoxon test. The overall time includes time spent backtracking and expanding parsing states, while the reference resolution time is only the time spent identifying reference mappings in a parsing state and grounding them to existing knowledge structures. This data is summarized in Table 6.3.

Figure 6.7 depicts the relative total time to read a text with the episode in memory normalized to the time spent reading a story for the first time. The relative differences are small, the overall mean is 1.08, and the median is 1.00.

In the overall data, the sections of the corpus which resulted in fewer parsing states or fewer reference mappings being evaluated fared slightly better when reading a text for a second time. Table 6.4 represents the same set of 125 texts presented in Table 6.2.
Table 6.3 Processing time when reading the extended corpus with and without existing episodic knowledge.

<table>
<thead>
<tr>
<th>corpus section</th>
<th>paraphrases</th>
<th>reference resolution time (ms) per sentence</th>
<th>total time (ms) per sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>first time reading</td>
<td>difference with existing knowledge</td>
<td>first time reading</td>
</tr>
<tr>
<td>election</td>
<td>6</td>
<td>29.1 26.7 8.5 4.2</td>
<td>246.1 133.3 4.4 6.3</td>
</tr>
<tr>
<td>geography</td>
<td>6</td>
<td>109.9 101.0 34.4 26.5</td>
<td>2084.0 1427.6 -43.9 1.6</td>
</tr>
<tr>
<td>history</td>
<td>33</td>
<td>63.5 20.0 23.3 5.0</td>
<td>473.2 192.5 8.3 5.0</td>
</tr>
<tr>
<td>lebanon</td>
<td>29</td>
<td>235.5 40.0 42.0 10.0</td>
<td>1797.2 726.0 11.1 0.0</td>
</tr>
<tr>
<td>military</td>
<td>14</td>
<td>131.2 30.8 40.0 -1.0</td>
<td>1176.6 442.5 58.8 -7.5</td>
</tr>
<tr>
<td>organization</td>
<td>4</td>
<td>42.8 6.7 19.2 3.3</td>
<td>786.8 130.0 32.3 6.7</td>
</tr>
<tr>
<td>person</td>
<td>10</td>
<td>77.9 15.8 41.6 5.7</td>
<td>1046.5 191.3 8.9 8.8</td>
</tr>
<tr>
<td>relation</td>
<td>10</td>
<td>144.9 40.0 16.4 3.3</td>
<td>905.4 291.7 -76.7 0.0</td>
</tr>
<tr>
<td>terrorism</td>
<td>13</td>
<td>61.1 31.3 -10.9 1.0</td>
<td>683.6 331.3 -132.3 -22.5</td>
</tr>
<tr>
<td>total</td>
<td>295</td>
<td>98.2 16.7 27.7 5.0</td>
<td>1051.0 300.0 -5.4 3.3</td>
</tr>
</tbody>
</table>

Figure 6.7 shows the relative total time per text for each story when the relevant episode already exists in memory. Horizontal line at 1.0 indicates the value of no change. Values below the line indicate a saving in time when the episode exists in memory. This graph includes values for all 295 texts.
<table>
<thead>
<tr>
<th>corpus section</th>
<th>paraphrases</th>
<th>reference resolution time (ms) per sentence</th>
<th>total time (ms) per sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>first time reading</td>
<td>difference with existing knowledge</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>median</td>
<td>mean</td>
</tr>
<tr>
<td>election</td>
<td>6</td>
<td>31.2</td>
<td>21.9</td>
</tr>
<tr>
<td>geography</td>
<td>6</td>
<td>133.3</td>
<td>134.1</td>
</tr>
<tr>
<td>history</td>
<td>33</td>
<td>41.7</td>
<td>30.0</td>
</tr>
<tr>
<td>lebanon</td>
<td>29</td>
<td>139.4</td>
<td>56.7</td>
</tr>
<tr>
<td>military</td>
<td>14</td>
<td>164.3</td>
<td>30.8</td>
</tr>
<tr>
<td>organization</td>
<td>4</td>
<td>56.3</td>
<td>50.5</td>
</tr>
<tr>
<td>person</td>
<td>10</td>
<td>73.7</td>
<td>19.0</td>
</tr>
<tr>
<td>relation</td>
<td>10</td>
<td>365.0</td>
<td>174.7</td>
</tr>
<tr>
<td>terrorism</td>
<td>13</td>
<td>74.9</td>
<td>33.3</td>
</tr>
<tr>
<td>total</td>
<td>125</td>
<td>97.6</td>
<td>25.0</td>
</tr>
</tbody>
</table>

Table 6.4 Processing time when reading the extended corpus with and without existing episodic knowledge for the 125 text subset depicted in table 6.2.

It can be seen that those that explored less ambiguity garnered greater savings in time. However, these stories also generally took longer to process in the first place, and so as a relative difference the savings are diminished. Comparison with Kendall’s method shows only a weak but significant correlation between reduction in parsing states and the total time (tau 0.46, p-value < 0.001). Although the shift was not large this shows that a reduction in parsing states and reference mappings evaluated can result in a reduction in processing time.

Figure 6.7 depicted the relative differences in time, another comparison would be to consider the absolute differences. Figure 6.8 is a histogram of the difference in total processing time per sentence between the two conditions. This plots the data represented in Table 6.3 in the difference with existing knowledge column under total time. This data follows a normal
distribution with long tails. Shown are the 214 texts (73%) that make up the outer quartile range. The median difference is 3.3 ms/sentence, and the inner quartile is between -14.2 and 22.6 ms/sentence.

Looking at only time spent performing reference resolution shows similar results. Figure 6.9 shows a histogram of differences in reference resolution processing time per sentence. The outer
quartile range of 235 texts (80%) are plotted. The median difference is 5 ms/sentence and the inner quartile range is between 0 and 20 ms/sentence. Again these differences are small, although significant.
There is no correlation between rate of reading and text length. The performance difference for having existing knowledge available is independent of the number of sentences; it is a fixed and relatively small (3.3ms/sentence) cost per sentence.

6.4.3 Discussion

The reduction of parsing states can be primarily attributed to cases where in the first time reading condition the best first search is on the correct path, and backtracks (erroneously) to consider other branches before returning to the correct branch. The re-reading condition performs better by being able to use the references to existing knowledge structures to know that it is on the right path and avoid the backtracking. Additional (and significant) reduction in the number of parsing states evaluated could likely be gained by altering the search algorithm to better leverage existing knowledge structures being tracked in the running interpretation, or being integrated by new interpretation fragments, in order to better prioritize the order in which new parsing states are explored.

The reference mapping algorithm processes an ordered list of mappings, and can leverage memory to know when it has found a correct mapping and stop its processing. In the absence of existing knowledge structures it uses a weaker heuristic for selecting a mapping that requires it to evaluate more mappings. This algorithm better leverages existing knowledge and this explains why the reduction in ambiguity was seen as significant here, but not in terms of number of parsing states.
The data from this experiment provides strong support for the first claim that the reference resolution algorithm in REDMAP can leverage existing episodic memory to reduce the amount of ambiguity that needs to be evaluated. However the results of this evaluation show that while REDMAP is able to exploit existing knowledge in order to evaluate less ambiguity, the run time cost to do so currently mitigate any gains. While this result is not entirely positive, it is reassuring that the overhead to interact with the knowledge base is relatively small, and does not compound with text length. These results demonstrate that the REDMAP algorithms are scalable when interacting with a large knowledge base. They also suggest that with improvement a reduction in reading time might be obtainable when existing knowledge is available.

Of the texts in the extended corpus 41 (14%) retrieved some bindings for the first time reading condition that had to be ruled out during reading. These 41 texts showed no significant difference in distributions from the full set of data when compared with a Wilcoxon test for the metrics evaluated in this section. There was a slight (and significant $p < 0.05$) shift in reference resolution time, but there was no shift in the distribution of overall processing time. Given the other data reported on here as to how the reference resolution algorithm performs, this would be expected. This data provides weak support for the claim that REDMAP is not significantly impacted by additional episodic remindings that must then be ruled out during reading. However additional investigation is warranted to verify the extent of this claim, in order to determine how many episodic remindings the typical story will produce after reading a very large corpus of similar content, and how many remindings are required before a significant impact is experienced on the reference resolution algorithm.
The data indicates that the current algorithms, leveraging episodic memory, nearly pay for themselves. And as discussed in chapter 4, regarding the reference resolution algorithm, there is plenty of potential room for improvement. Additional investigation is definitely warranted based on these results. It should also be noted that this algorithm produced representations that are fully grounded in the existing episodic memory, and to complete the integration only the new assertions need to be added. There is no subsequent work to be done in order to integrate with memory. It is hard to provide a direct comparison of what this savings is, but this algorithm also provides additional savings in this regard as well.
Chapter 7  Related Work

The key topic of the work presented in this dissertation is a method for natural language understanding that:

1. produces representations that are fully integrated with an underlying episodic memory,
2. uses episodic memory as it parses to influence decisions, and
3. manages the resolution of all ambiguity as the process of resolving references to underlying episodic memory.

Each of these features corresponds to a position along dimensions that can be used to compare related work, with the following three questions. What is the model for understanding and the final output of the parser? How integrated is episodic memory with the language understanding process? How is ambiguity and reference resolution handled by the parser? Related work along each of these dimensions is presented in the rest of this chapter.

The decision of what approach to use for language understanding depends heavily on what the goals of the understanding system are. For example, is the system intended to form complete understandings of sentences or paragraphs, and are these understandings to be integrated with episodic memory? Or is the system designed to gather only a few facts selectively from large bodies of text? This chapter presents a range of approaches for language understanding.
A primary component of the work presented in this dissertation is its approach to reference resolution. Approaches to reference resolution can be divided broadly into those that use structured, semantic knowledge and those that do not. The vast majority of existing work on reference resolution looks at connecting one anaphoric reference to another textual reference in the same text. This approach is frequently employed by pipeline systems and is typically performed early in the pipeline before episodic memory is available, and frequently even without deep semantic structures. In contrast there are knowledge-based approaches to reference resolution that are more diverse in their methods, and these approaches will be highlighted throughout the chapter.

### 7.1 Pipeline Parsing

The most common approach for language understanding is to decompose the problem of deep understanding into a series of modular components formed into a *pipeline*. As architecture for developing software this approach is very attractive to the computer scientist. Modularity allows independent research on individual components and for the research to approach a solution incrementally for language understanding. Since integration with episodic memory is typically seen as one of the last stages of the pipeline, work can be done on pieces early in the pipeline while other researchers are still figuring out how episodic memory is structured and what is stored in it.
7.1.1 Reference Resolution in Early Pipeline Stages

Much of the work in this arena is focused on early stages of the pipeline such as: named entity recognition, part of speech tagging, and identifying syntactic structures. These stages are also where the majority of research on reference resolution in the context of pipeline understanding is performed. The research here, by design, only addresses a subset of ambiguity. This is summarized by Mitkov (2002)

The vast majority of systems only handle nominal anaphora since processing anaphors whose antecedents are verb phrases, clauses, sentences or sequences of sentences is a more complicated task.

The majority of projects focus on pronoun resolution; a good number of projects also address resolution of definite descriptions (some including indirect anaphora) or zero pronouns. However, the resolution of non-nominal anaphora, which is arguably more difficult to handle, is almost completely ignored.

In essence, most work is on one type of one kind of reference – nominal anaphora. In addition to only considering nominal anaphora, most anaphora resolution research only attempts to identify a previous textual segment to connect an anaphor not a semantic or episodic structure to which the reference is actually grounded.

Early work on heuristic anaphora resolution ranged from using simple heuristics in STUDENT (Bobrow 1964) to more elaborate ones in SHRDLU (Winograd 1972). Other work, such as LUNAR (Woods 1970) adds some shallow semantic information to the idea of using heuristics.
The Hobbs (1976) so-called naïve approach is among the most commonly used, and operates over syntactic parse trees. The BFP algorithm (Brennan, Friedman, and Pollard 1987) uses centering (Grosz and Sidner 1986) and SPAR (Carter 1985) uses focusing (Sidner 1979) to resolve anaphora.

RESOLVE (McCarthy and Lehnert 1995) used machine learning techniques to constructed C4.5 decision trees that classify coreferent noun phrases. Mitkov’s self described “knowledge poor’ technique MARS (Mitkov 2000) focus on anaphora resolution using only part of speech tagging and noun phrase extraction. It is motivated by the assumption that leveraging knowledge is too expensive.

7.1.2 Discourse Representation

Pipeline processors that go beyond shallow sentence processing typically do so by adding a discourse model that organizes multiple sentences into a coherent structure. Grosz et. al. (1995) present work on producing coherent discourse structures over multiple sentences. One of the more commonly used discourse models is DRT (Discourse Representation Theory) (Kamp and Reyle 1993; Poesio 2000). DRT represents information in Discourse Representation Structures (DRS) that contain a set of discourse referents and a set of conditions that express properties of those referents, including how they are connected. As parsing progresses new information is incorporated to existing DRSs when possible, otherwise new DRSs are created as needed.
Some linguists argue that discourse structures play a dominant role in reference resolution. Cornish (1986) suggests that the antecedent of an anaphor should be viewed as a discourse structure in memory, rather than a piece of text. Cornish refers to what is commonly thought of as the antecedent as an antecedent-trigger which merely primes concepts for being referred to, but is not that which the anaphor actually refers. He also uses the concept indexical segment to mean essentially the context of the anaphor. With that terminology Cornish makes the following strong claim as to how anaphors are processed.

… the anaphor, within the context of the indexical segment as a whole as an effective utterance, itself determines its potential reference value – the actual referent being yielded via the integration of the indexical segment as a whole with the discourse context constructed up to that point. The possibility of pairing the anaphor with an antecedent-trigger expression occurring within the co-text is only feasible post hoc – the outcome or symptom of the anaphora expressed, rather than being its cause.

Cornish’s account for anaphora argues that the process of understanding at the discourse level is the process of resolving references.

### 7.1.3 Pipeline Systems in Use

Commonly used pipeline parsers include Allen’s parser (1995), Charniak’s parser (2000), the Link parser (Sleator and Temperley 1993), and the Stanford parser (Klein and Manning 2002). Powerset (Converse et al. 2008) is a state of the art pipeline processor built on PARC’s XLE parser (Maxwell and Kaplan 1993). It can produce detailed sentence level understandings of much of the content of the Wikipedia (Wikipedia). Powerset can resolve some anaphora across sentences, such as pronouns, however for the most part it, like most other pipeline parsers that
are not integrated with a deep semantic memory, does not perform understanding across sentences. For example, a multiple sentence description of an event, like an election, is not formed into a single coherent structure in the representation produced by these parsers.

7.1.4 Discussion

The work presented in this dissertation aims to produce understandings that are fully integrated with underlying episodic memory. It does so without producing an intermediate syntactic analysis, therefore direct comparisons to other work that only produce syntactic structures is neither salient nor possible. Most of the pipeline-based approaches for reference resolution focus primarily on pronoun resolution or noun-phrase to noun-phrase mappings, by connecting spans of text. All references are seen as equal to REDMAP and processed by a uniform mechanism, whether the reference is generated by a noun-phrase, or is an event implicit in a verb-phrase. When performing reference resolution across sentences the systems described above operate with information at the discourse structure level or lower. REDMAP performs all of its reference resolution operations with semantic structures, ones more deeply connected to underlying memory than those used even by DRT, which are typically lexically motivated. REDMAP is capable of making decisions based on the contents of its semantic and episodic memory. Although there is a token similarity between DRT and REDMAP in that they can be both described as collecting references and constraints on those references as they parse, what each system views as a reference and as a constraint are radically different. The biggest difference being that REDMAP grounds all references to episodic memory whereas DRT is typically applied prior to memory integration.
7.2 Semantic Understanding

When I refer to semantics in a language understanding system I consider a level of structured representation at least at the level of Conceptual Dependency (CD) (Schank 1969) that constructs semantic meaning from text above the lexico-semantic structures constructed by something like HPSG parsers (Pollard and Sag 1994), where semantic structure is closely tied to linguistic structure.

7.2.1 Early Semantic Understanding Systems

Earlier systems that output deep semantic structures included systems that took a top down approach to language understanding starting with Spinoza, Spinoza II (Schank, Tesler, and Weber 1970), and MARGIE (Riesbeck 1975) which understood sentences by setting expectations for what would come next. The MARGIE parser used if-then rules called requests to construct meaning forms. For example, requests attached to “give” would look for the words following to describe a physical object (e.g., “give a book”), or an action (e.g., “give a kiss”) in order to produce completely different representation of the meaning of the sentence.

Word Expert Parsing (Small and Rieger 1982) took the idea of requests even further by allowing them to be small programs that could do almost anything. Other early understanding systems capable of outputting deep semantic structures included SOPHIE (Brown and Burton 1975), and work by Hendrix (1975). All of these early systems were typically ad hoc and domain specific, organized around collections of phrasal patterns, for instance, what one would eat, or why
someone would pick up a stick. Commitments were not made in these systems to consistent semantics across domains that is necessary to produce systems that can function in multiple domains or leverage knowledge across domains.

### 7.2.2 Standardizing Semantics

MARGIE performed understanding in two stages, a parser, called an *analyzer*, would operate first to identify concepts in the text (such as a reference to an action), and an *inferencer* would then apply its if-then rules to interpret the text. The idea of a parser that could produce CD structures was extended with better control structures and standardized requests in systems such as ELI (Riesbeck and Schank 1976), ELI-2 (Gershman 1979), and CA (Birnbaum and Selfridge 1979).

The ideas behind the inferencing stage of MARGIE were extended in a series of systems that used one of the previous parsers as their front end. Where MARGIE attempted to understand sentences, SAM (Schank and Abelson 1977) formed understandings by mapping text to generalized event descriptions called *scripts*. PAM (Wilensky 1978) and POLITICS (Carbonell 1979) understood stories in a similar fashion but had knowledge structures for plans and goals. These systems could understand complex events and perform reference resolution by leveraging their semantic knowledge. For example, in reading the following classic restaurant story, the analyzer would output, for the third sentence semantics equivalent to `<male-human> ordered <hamburger>`.
John went into a restaurant.
The waiter came over.
He ordered a hamburger.

When SAM merged this representation with the script, because it is specified who does the ordering, SAM knows that “he” refers to John.

FRUMP (DeJong 1979) was the first parser in this line of research that did not attempt to fully understand all of the text it was given. Unlike SAM that attempted to fit every piece of text into the script, once FRUMP identified what script, if any, applied to a text it would scan for patterns connected to limited number of specific slots in the script.

Systems, such as IPP (Lebowitz 1980) and BORIS (Dyer 1983), built instances of complex concepts and stored them in memory. IPP was capable of refining concepts to through a hierarchy by attempting to find more specific concepts that applied to an input. BORIS would accomplish a similar task by issuing queries to memory for more specific types of concepts and creating them if they did not exist.

RESEARCHER (Lebowitz 1983) was a system that could resolve ambiguous references by appealing to its knowledge of objects in semantic memory. RESEARCHER was implemented in the domain of patents for computing equipment. In the phrase “a metal drive cover” it could disambiguate whether “metal” was modifying the drive or the cover by appealing to its existing knowledge of disk drives.
7.2.3 Modern Semantic Understanding Systems

The Explanation Agent (EA) (Kuehne 2004) is a modern semantic understanding system, implemented on top of Allen’s chart parser, that builds Qualitative Process Theory (Forbus 1984) descriptions in CycL of physical phenomena from multi-sentence texts. Tomai (Tomai 2009) extended this system by integrating DRT and added pragmatic constraints and abductive reasoning related to narrative structure in order to read texts such as fables. Both versions of EA build new semantic representations and do not refer to existing episodic instances in their interpretations other than named entities.

7.2.4 Discussion

These semantic understanding systems have access to existing semantic structures, such as scripts for quickly understanding the causality of common multi-event sequences. These structures also helped resolve reference efficiently. ResearchCyc does not provide a comparable set of structures and therefore REDMAP required more development in the area of reference resolution. On the other hand, all of the systems described in this section are build-and-store parsers; they do not leverage or remember their existing episodic memory. Unlike REDMAP these systems are not capable of recognizing that they have existing episodic knowledge for a given event. Every single time John walks into that restaurant SAM will assume it is a completely new episode. REDMAP will be reminded of existing episodes in memory as it parses and can extend those instances with new knowledge as it is provided. Although EA produces deep semantic representations in the semantics of ResearchCyc, unlike REDMAP it does not integrate its representations with existing episodic memory. Dehghani (Dehghani et al.
2008) however has used analogy (Falkenhainer, Forbus, and Gentner 1986) to compare EA-produced representations to existing episodic instances in memory.

### 7.3 Memory Based Parsing

Episodic memory plays an important role in REDMAP’s processing. The previous section looked at systems that could produce new episodic instances as a result of parsing. This section looks at systems that can do not only that, but that are also capable of using structured symbolic episodic memory during parsing in order to influence decisions.

#### 7.3.1 Early Semantic Understanding

The ideas underlying memory-based parsing date back to Quillian’s (1968) Teachable Language Comprehender (TLC). In TLC semantic interpretation came first, and syntactic filters were then applied to confirm that a given semantic hypothesis was correct. TLC used a marker passing algorithm on a memory that was a network of concepts. When a phrase like “lawyer’s client” was to be interpreted, TLC would place markers on the \textit{Lawyer} concept and the \textit{Client} concept, and the markers were spread until they intersected. In this case they would intersect at the \textit{Employs-Professional} concept. The path between the concepts is then viewed as a candidate interpretation. Concepts were annotated with syntactic filters, called \textit{form tests}, which would then be applied to verify the candidate interpretation. In this example there would be a form test on the \textit{Employs-Professional} concept that would expect the \textit{Professional} branch (in this case \textit{Lawyer}) of the path to come from text that is possessive. This particular form test will also
expect the two words to be adjacent. If the form test is satisfied then the interpretation is confirmed.

7.3.2 Direct Memory Access Parsing (DMAP)

DMAP (Riesbeck and Martin 1986; Martin 1990) searches memory for existing episodic knowledge as it parses. It approaches understanding as a recognition process, and only constructs new knowledge structures when it cannot find them. So fundamental is this view that the first implementation, DMAP-0, could only recognize existing knowledge. It could not add structures to memory.

DMAP’s recognition is done by mapping phrasal patterns to structures in memory. PHRAN (PHRasal Analyzer) (Wilensky and Arena 1980) incorporated the idea of phrasal patterns to direct memory parsers, but it was not until DMAP that Quillian’s ideas were realized in a functioning NLU system. Unlike TLC, all implementations of DMAP built thus far give up the ability to search for connections in a purely semantic way. In DMAP the form tests are intimately connected to the marker passing or pattern matching algorithms. This allows more complicated patterns to be leveraged and greater scalability. Functionally the pattern matcher in DMAP is similar to a tabular chart parser (Kay 1996), except its outputs are fully grounded in the underlying knowledge base. DMAP was applied to several targeted domains by Martin (1990) and Fitzgerald (1994).
7.3.3 Other Memory-Based Systems

The ideas behind DMAP were extended to build a translation system DMTRANS (Tomabechi 1987). DMTRANS operated by having patterns from multiple language connected to concepts in memory. Concepts were activated in one language and then generated in the other. DMDialog (Kitano 1990) was another system developed along the same lines, where translation was begun before utterances in the first language were complete in order to better facilitate dialog systems. Kitano and Moldovan (1992) also explored ways in which memory-based parsing could be accelerated by parallelizing it in hardware.

Other related work includes Charniak’s (1983) marker passing system, inspired by TLC, which searched a frame-based memory for potential interpretation(s) of the text. It used syntactic information and made other inferences to filter what was retrieved. This system would then produce a logic form to be stored into memory.

Fitzgerald also developed a new method for memory-based parsing for use in domains that did not already have well organized and populated knowledge bases connected to them. Indexed Concept Parsing (ICP) (Fitzgerald 1994) treats understanding and constructing meaning from text as a Case Base Reasoning (CBR) problem. For each segment of text to be understood ICP produced a vector of all the semantic concepts each of the words could represent. This vector was then used as an index into a case base of related meanings which could be applied to the text.
7.3.4 Discussion

While REDMAP is an implementation of DMAP, which is in turn heavily inspired by TLC, REDMAP has some significant differences from its ancestors. The systems described in this section update their interpretations after each word is processed. The results in section 6.1 showed REDMAP does not scale when searching memory incrementally at the word level. It updates only after each sentence. This difference is related to the fact that REDMAP is deployed on top of RearachCyc, which is significantly larger than memories used by earlier systems. Earlier implementations of DMAP, similar to the systems described in section 7.2, had access to knowledge structures, such as scripts, that not only focused search in memory, but also supported reference resolution. This type of knowledge is not currently provided in ResearchCyc and its absence necessitated new reference resolution algorithms being implemented in REDMAP. The new reference resolutions algorithms (see chapter 4) use memory to assist in deciding how two references across sentences can be connected. Although this is a difference with respect to DMAP implementation, in some respects this is a similarity to TLC.

7.4 Non-Symbolic Memory Network Models

The memory-based systems discussed in the previous two sections are built on top of symbolic memory. This section presents approaches to language understanding that use non-symbolic memory. There has been a range of research on connectionist and spreading activation models for language understanding. Of particular interest to this dissertation are models that pay attention to ambiguity or references.
Reviews of early connectionist (Rumelhart and McClelland 1987; McClelland and Rumelhart 1987) language understanding work can be found in Selman (1989), Diederich (1990), and Sharkey (1990). These systems were built using multiple layers that performed their processing in parallel and feedback into each other. A good example of this work is by Small et. al. (1982) that had three levels. The first level handled lexical information like morphology, the second identified word senses, and the third dealt with the connections between predicates and objects. The connections between layers would allow decisions in one layer to feedback into another layer, reinforcing or reducing activation, and therefore decisions, at that level. Reilly (1984) provides an extension to this work that adds another level to the model for anaphora resolution.

Kintsch (1988) and ACT* (Anderson 1983) augmented spreading-activation networks with conditional constraints, by connecting to an external symbolic system. ATLAST (Granger, Eiselt, and Holbrook 1986) is an early spreading activation model for language understanding that focused primarily on disambiguation and functioned by processing syntax and semantics concurrently. TOPIC (Hahn and Reimer 1983) used linguistic cues to adjust activation levels in the network in order to focus the activation from a set of anaphoric references to single nodes.

Spreading activation models perform reference resolution by using relative levels of activation in the network to make decisions about the referents of new inputs. ROBIN (Lange and Dyer 1989; Lange 1992) is a connectionist model that can perform some high-level inferencing and disambiguation within the spreading-activation system. REMIND (Lange and Wharton 1994) is
an extension of ROBIN which also leverages connections to episodic memories. These episodic memories are activated and retrieved as the language is being understood.

7.4.3 Discussion

All of the systems described allow semantic and episodic information encoded in their memory networks to influence parsing decisions. These systems have the advantages of being efficient by the nature of being massively parallelizable, and of being robust to noise or novel inputs. Network-memory systems however have difficulty when interacting with episodic memory. Many early systems have difficulty dealing with input that discussed multiple of the same type of thing, a limitation not found in REDMAP. REMIND simultaneously searches semantic and episodic memory as it parses. In this regard it is similar to DMAP in its goals. However, REMIND only finds the closest memory (or set of memories) activated. It is unable to identify whether the exact episodes found are the ones being referred to, or only the closest known memories. REMIND also does not have the ability to automatically identify and segment out the information pertaining to a new episode in memory. In order for a memory to be retrieved later the new episodes must be manually added.

7.5 Information Extraction

When it comes to acquiring knowledge from text, this dissertation focuses on understanding all of the content being communicated in a single passage of text. Other approaches, such as those used by the Information Extraction (IE) community, focus on acquiring select facts from large
collections of documents. Information extraction approaches use both statistical and phrase-based parsing. Most systems that learn by reading are aimed at extracting particular kinds of facts from the web. For example, KnowItAll (Etzioni et al. 2005) extracts named entities and OPINE (Popescu and Etzioni 2005) extracts properties of products. While much of the IE work focuses on entity detection and relation extraction from isolated sentences, there has been work to build case frames starting with PALKA (Kim and Moldovan 1993) and AutoSlog (Riloff 1993). Humphreys et al. (Humphreys, Gaizauskas, and Azzam 1997) focuses on event references.

For the most part Information Extraction systems learn new patterns to identify information but these systems do not typically represent their knowledge in a deeply structured memory or ontology. The concepts and relations they extract generally have minimal representation behind them and lack, for example, knowledge about what a mayor or a city is, or the relationships to facts being gathered by other extractors. Carlson et al. (2009) has incorporated ontologies and semantic constraints with sets of extractors so that they can integrate their information into larger knowledge bases, and improve accuracy by training multiple extractors together. For example, in Mitchell’s work, extractors know that baseball players are on teams, and teams play other teams, and have a home city, etc. These constraints are used to assist the learning. Research is also being done to learn new axioms from the data to extend the ontology.

Brussell (Wagner et al. 2009) builds understandings of complex and evolving events from multiple documents read over time. Brussell shares traits with FRUMP and Information
Extraction systems. It does not try to understand whole documents; instead it selectively extracts information to instantiate scripts. It merges information from multiple documents about the same event into one script instance in memory. Brussell uses voting methods to resolve conflicts when several values for a given script filler are gathered from multiple documents. In this way it resolves references across multiple documents and builds instances in episodic memory.

7.5.1 Discussion

REDMAP is different from IE systems as it reads single passages and does not extend its vocabulary. Nor does it actively manipulate its ontology by learning new categories or properties of those categories, as some IE systems, such as Opine, are capable. REDMAP and the work by Mitchell do share one idea though, in that they both actively evaluate during reading how new information will fit in with what is already known, and use this to influence learning. Since Mitchell’s extractors operate in an ensemble, the language patterns being learned effectively have semantic type constraints, similar to DMAP patterns. However their semantics are evaluated after the pattern matches to check the validity of the output, not before in order to support the matching, as does REDMAP.

Brussell integrates the episodic memories it creates and maintains with ResearchCyc, the same knowledge base REDMAP uses. Brussell, unlike REDMAP, does not use episodic memory or its existing knowledge to assist in extraction of information from the text, beyond detecting proper names such as people and places. In contrast to REDMAP, Brussell can extract information from a wide range of lexical structures, however it can only understand content for
which it has a script. REDMAP on the other hand can piece together descriptions of events it has never read about before.

### 7.6 Other Methods for Reference Resolution and Reminding

Cassimatis (2008) discusses performing reference resolution as an inference and constraint satisfaction problem. This approach combines constraints from the language and from real-world knowledge into a maximum a posteriori inference (MAP) problem. Clark and Thompson (2009) has taken interpretations for multiple paragraphs on the same topic and combined them to form summarizations in a somewhat similar approach. Interpretations are built for each sentence using BLUE (Boeing’s Language Understanding Engine) (Clark and Harrison 2008). The interpretations are then forward chained through DIRT (Discovery of Inference Rules from Text) paraphrase rules to create a set of possible meanings and implications for a sentence. All the assertions and mappings are then fed into a satisfiability solver (MAX-SAT) which attempts to find a configuration that produces maximal overlap. In doing so a summary is produced and references are resolved across paragraphs. It is an open question as to how this approach could be used to merge into or retrieve from existing episodic memory.

DMAP attempts to treat language understanding as a problem of recognition and retrieval of existing episodic memory. Other more general models for retrieving existing episodes from memory exist, and the following are a few examples. ARCS (Thagard et al. 1990) is a multi-stage model for retrieval from memory that first sets up a network of concepts related to the source, and then uses a spreading activation algorithm to identify a target memory. MAC/FAC
(Forbus, Gentner, and Law 1995) is another multi-stage model that starts with a source representation, and constructs a feature vector that it then uses to identify several candidates efficiently from a case library. Those candidates are then compared via analogy to the source to identify the best reminding from episodic memory. Finally, work by Nuxoll and Laird (2007), motivated by Case Based Reasoning (Kolodner 1993), is investigating episodic memory in Soar (Newell 1990). This system functions by taking snapshots of working memory which are later retrieved for use when similar working memory circumstances are encountered.

7.6.1 Discussion

The approaches proposed by Cassimatis and Clark, look at large quantities of constraints simultaneously and leverage the SAT-solver’s ability to converge on a correct answer efficiently. Similar to the reference resolution algorithm described in chapter 4, Clark’s work hypothesizes mappings between references. However, it proposes all mappings exhaustively, while the algorithm in chapter 4 attempts to filter this list using constraints from the ontology. REDMAP uses semantic and episodic memory to guide the reference resolution process incrementally. Furthermore, future work for REDMAP (see the next chapter) points at ways this memory could serve as an even better guide to direct the process. It is an open question how constraints from episodic memory could be efficiently added to the other approaches in a way that can scale.

The three systems that support episodic retrieval require semantic representations to be built prior to interacting with memory, whereas REDMAP uses interacting with episodic memory in order to build the semantic structures in its representations. REDMAP’s remindings are also
retrieved incrementally starting where it left off previously. The above models could probably be easily modified to support this, however that is not how they are currently designed.

## 7.7 Language Understanding by Humans

While REDMAP is not attempting to model human functioning in any way, it does attempt to accomplish a human process, language understanding. The way that humans understand language can provide valuable insight to the plausibility of a given approach. At the very least, we know the methods and processes employed by the human are sufficient for understanding language, even if they are not necessary or the only approach. They are however, currently, the only known successful approach and therefore understanding them is worthwhile.

A two stage, syntax then semantics, model of human language understanding has had support from Grice (1975), Fodor (1983), and many others. This model corresponds to at least a simple pipeline model, where syntax is processed before semantics are considered. However, before the introduction of neuroimaging, it was difficult to isolate when specific mental processes are occurring.

Only recently have researchers through neuroimaging been able to investigate this claim experimentally in humans. Camblin et. al. (2007) illustrates the depth of semantic knowledge humans bring to bear early in language understanding. This work suggests that humans are activating semantic information early in language understanding, at the same time as syntactic information. Furthermore, the semantic information is not just shallow understandings, such as
sweaters are itchy, but very detailed information, such as what body parts sweaters can make itchy.

Hagoort and van Berkum (2007) go beyond claiming that a one-stage model of language understanding is plausible, and further assert that given the evidence a pipeline model is not plausible in humans.

In psycholinguistics, this analysis of meaning has evolved into the standard two-step model of language interpretation, according to which listeners (and readers) first compute a local, context-independent meaning for the sentence, and only then work out what it really means given the wider communicative context and the particular speaker. We have discussed a wide range of ERP and fMRI findings that collectively do not sit well with this two-step model. Instead, the findings consistently point to a one-step model of language interpretation. Not only core linguistic information about the phonology, syntax and semantics of single words and sentences, but also discourse information, world knowledge and non-linguistic context information immediately conspire in determining the interpretation of compound expressions.

Based on this evidence for a one-step model of human language understanding, it is reasonable to explore building systems that can leverage both syntactic and semantic information early in the language understanding process. Investigating models such as DMAP and systems that can bring large amounts of deep semantic and episodic memory to bear early in language understanding is well warranted.
Chapter 8  Conclusion

I conclude this dissertation with a summary of what has been presented, a discussion of future work, and some final remarks.

8.1 Summary

Understanding language requires relating what has been said to existing knowledge. This requires identifying and grounding references in the text to existing semantic and episodic structures. Reference resolution is the process of identifying the concepts in memory to which the language is referring. In this dissertation I presented a method for language understanding that treats all ambiguity resolution as a problem of reference resolution: grounding references to episodic memory. This is different from traditional approaches to language understanding in two key ways. First, memory integration is an intrinsic part of the language understanding process beginning at the most early stages. Second, this approach, like all DMAP approaches, makes language understanding fundamentally a problem of memory recognition.

In order to evaluate this idea I built an implementation of DMAP called REDMAP (Reference resolution in Episodic memory for DMAP) that resolves references dynamically and grounds them to existing knowledge structures as text is processed. The key ideas are, first that episodic memory can be used to facilitate connecting one reference in the text to another, and second that adding episodic memory is adding constraints that reduce ambiguity not multiply it.
Performing this kind of understanding, grounding references to existing semantic and episodic knowledge structures on a large memory, raises serious questions regarding the scalability of this approach. A primary contribution of this dissertation was to addresses these concerns by evaluating the scalability of REDMAP. In this dissertation I provided support for the following four claims.

**Claim 1:** A memory-based reference resolution algorithm (REDMAP) can provide broad coverage of and extending an existing large knowledge base by grounding to existing episodic memory as it parses and using that memory to reduce ambiguity.

Estimates on upper bounds of coverage and recall were presented in section 5.1 indicating that REDMAP can cover nearly all the collections in ResearchCyc. Coverage of predicates is significantly lower (improving this is addressed as future work in the following section).

Measurements of quality were focused primarily on precision over recall, section 5.4 showed that REDMAP-produced assertions were both accurate and useful to automated question understanding. Section 5.2 discusses the precision of REDMAP output overall.

Sections 6.3 and 6.4 provide clear evidence that REDMAP is able to leverage existing episodic knowledge structures to reduce the number of reference mappings evaluated when merging the references in a new sentence with the running interpretation.
Claim 2: The reading rate (mean time per sentence for a text) of the REDMAP algorithm is empirically independent the number of references in the text and the length of input (number of sentences).

Support for this claim is provided throughout chapter 6, primarily in section 6.2. While the number of references is increasing with each sentence read, and REDMAP is grounding all references to episodic memory thereby adding yet another level of ambiguity to resolve, the empirical analysis shows REDMAP does not get slower and slower as it reads. Although it was hoped that episodic memory could be leveraged to accelerate the rate of reading, this stronger version of claim 2 was not shown. Instead REDMAP pays a slight overhead in reading time for leveraging episodic memory, discussed in section 6.4. However, this is a fixed, and relatively small, cost per sentence. Furthermore in paying this cost REDMAP is able to simultaneously perform the task of episodic memory integration, a step left for subsequent processes or even future work in most other NLP research.

In addition to the preceding two strong claims I provided weaker support for the following two claims.

Claim 3: Complexity of syntax is not a predictor of processing time in REDMAP.

The third claim was a surprise outcome from the experiment presented in section 6.3, designed to test the hypotheses behind claim 2. This research sought to evaluate the role of memory in language understanding, and the larger project, the Learning Reader, aimed to read and reason about a broad range of content in ResearchCyc. To both of these ends we decided to simplify the
input provided to REDMAP to avoid encountering problems stemming from syntactic complexity. The results in section 6.3 show that oversimplification led to increased work in some cases for REDMAP when compared to other more syntactically complex sentences. It is still an open question as to how much syntactic complexity REDMAP can handle.

**Claim 4:** In domains that have fewer or less specific patterns, REDMAP can piece together partial understandings, at a cost to the accuracy and quantity of assertions produced. This relationship can be predicted to some degree from the amount of text understood.

Section 5.3 presented support for the fourth claim. The data in this section showed that in domains that REDMAP had weaker pattern coverage, quality and quantity of assertions produced could be controlled by adjusting the amount of partial sentence matching allowed. When more coverage is required quality is improved however the quantity of assertions is decreased.

### 8.2 Future Work

Future work possibilities range from improving the basic properties of REDMAP including coverage and scale, to reading about generalizations, and even to developing new models of episodic memory.
8.2.1 Increasing Coverage

While the linguistic information extracted from ResearchCyc provides direct coverage for the majority of collections, a refinement algorithm needs to be implemented to be able to generate reference to other collections, not just recognize them. Refinement is the process where more specific knowledge structures can be produced when only general linguistic references are made. For example, when a professor speaks to a class it could be encoded as speaking, but a more specific type of speaking, lecturing, is more accurate. Refinement is discussed in this dissertation in section 5.1.1 and in the original DMAP work (Martin 1990).

In stark contrast to collection coverage, predicate coverage is very low. The most direct way of increasing this coverage is to add additional linguistic information to ResearchCyc for the missing predicates. This could be a large effort as there are currently over 8,000 uncovered predicates, requiring a seven-fold increase in the number of linguistic annotations in ResearchCyc for predicates. Although this would be a large and labor intensive project it is a tractable one. However, over 6,500 of the predicates in ResearchCyc are never used in a single top-level assertion. If the coverage of unused predicates is not required, then the number of linguistic annotations only needs to increase three-fold.

It is possible other existing resources could be leveraged to facilitate the effort, such as integrating the information contained in VerbNet (Schuler 2005) in order to automate generating some of the patterns. The methods employed by the Information Extraction community (see section 7.1.5) might also be leveraged to acquire linguistic patterns. The predicate names or
comments in ResearchCyc could potentially be mined for information, for example, with some adjustment, assuming predicates represent the English verbs in their names. More ambitious than looking for patterns to leverage in predicate names or comments would be to develop a system capable of reading and understanding the kind of generalized information in the comments in ResearchCyc or a dictionary to bootstrap itself. This would require the ability to understand generalizations, discussed later in section 8.2.4.

8.2.2 Improving Performance and Scalability

As with the previous implementations of DMAP, parallelization is a likely source of improved performance. The algorithms used in REDMAP (see section 4.2.1) for both exploring new parsing states and for generating reference mappings should be parallelizable. This would allow REDMAP to evaluate more ambiguous options, or arrive at decisions sooner. Additional improvements to the reference resolution algorithm include exploring building mappings incrementally. Currently partial mappings may be being evaluated redundantly (see section 4.5), and incrementally checking against episodic memory may be more efficient.

The pattern matcher in REDMAP ignores nearly all syntactic constraints. This causes increased perceived ambiguity when patterns are allowed to match erroneously, for example over phrasal boundaries. Incorporating syntactic information would allow REDMAP to better prioritize its search, if not filter erroneous matches outright. There is a large area to explore regarding combining syntax and semantics, varying from giving semantics absolute priority, as in REDMAP, to giving syntax priority, as in pipeline models, and many methods for combination
in between. Along the same lines, the reference resolution algorithm in REDMAP also ignores linguistic cues. For example, determiners or words like “another”, “different”, or “same” that provide information about what references are eligible for merging and when a reference is likely to be to something yet to be mentioned in the text.

The coherence metrics discussed in section 5.2.2 give REDMAP a way of recognizing when it is producing an unconnected interpretation. It is possible this information could be used introspectively to identify either when the system is producing an error, or possibly as a cue for additional, more expensive reasoning. For example, an unconnected interpretation might indicate that a metonymic reference (not currently handled by REDMAP) was made in the text, and that additional reasoning should be done to figure out what references are related.

8.2.3 Increasing the Role of Memory

As seen in the types of information extracted from ResearchCyc in chapter 3, and discussed in section 4.4.1, REDMAP has no patterns for concepts expressed with language larger than a sentence. One of the primary operations in REDMAP is figuring out how the references in one sentence are connected to those in another. Multi-sentence patterns that would map to larger concepts in memory such as scripts or MOPs would greatly reduce the amount of work necessary. It would also then enable the exploration of the methods used in REDMAP for connecting even larger pieces of knowledge, for example the reasons behind a retaliation, versus the lower level knowledge of who performed the attack.
Additional research should also be performed to see if episodic memory can be used to produce better expectations, not only for reference resolution but also for exploring parsing states in general. The hypothesis is that episodic memory can be used to actively guide search, not just prune it as it is currently implemented. Taking these ideas one step further would be figuring out how to leverage related remindings, not just exact remindings. For example, if a text is discussing a bombing in Al Anbar that is new to the system it should be able to leverage its knowledge about other bombings that occurred, for example in Baghdad, to help understand the current text. For example, the other similar bombings will have been performed by insurgents or terrorists, something was damaged, and it was likely either a roadside bomb or a suicide bomb, etc. All of this information can provide cues for the reference resolution algorithm. For example, if a road is mentioned it should not have to iterate though all the possible references to which the road could be connected. The system should be able to look at similar events and hypothesis that the way a road appeared in the representation of the previous event is similar to how it will be used in the new event. For example, it appears in certain relations specifying the location of a bomb, or the path of a convoy, etc.

8.2.4 Reading about Generalizations

This dissertation focused on reading about instances of people, places, and events. A system that is capable of learning new concepts by reading requires the capacity to read about and manipulate its own ontology and semantics. It will need to learn new collections as well as relations. This requires reading sentences such as “Elephants like peanuts” and knowing that it is about elephants in general and not a specific elephant. Generalizations are encountered in direct
statements such as, “Rabbits are lagomorphs,” requiring a reader unfamiliar with taxonomies of animals to infer something about the meaning of rabbits and a new word, “lagomorphs.” Texts are also frequently written in ways that intermix instance-based and generalized knowledge, such as, “This rabbit is a lagomorph.” Often a concrete example is used when discussing a complex piece of generalized knowledge. Or generalized knowledge might just be interleaved as in T1. An intelligent reader must be capable of not only discerning these two contexts, but also updating one and using it to understand the other simultaneously.

**T1**
Clyde is an elephant.
Elephants like peanuts.

There is a lot of generalized knowledge in ResearchCyc encoded in axioms that would require further investigation to figure out how to leverage and manipulate it with REDMAP. Other generalized knowledge, such as collections or assertions represented with higher-order predicates, can be represented and reasoned about in the semantics of ResearchCyc just as directly as instance based knowledge. However, REDMAP makes some simplifying assumptions regarding dealing only with instances. For example, REDMAP assumes “rabbits” is a reference to some group of rabbits, but not all rabbits in general. Extending this would add ambiguity at the pattern matching level. With respect to reference resolution, REDMAP would need to be capable of considering references to instances and generalizations simultaneously, such as those in T1. It would also need to be able to maintain those references consistently across the interpretation, knowing which assertions are related to generalizations and which belong to instances or groups. Additionally understanding all the examples presented in this
section requires the ability to discuss prototypical instances and abstract their properties to the class as a whole. Finally, REDMAP would need to deal with ensuring the consistency of generalized knowledge being learned or changed with what has already been read in a text, as well as with episodic memory. Validating or comparing new generalized knowledge with the contents of episodic memory could be combinatorially expensive, if not prohibitive with some memory organizations.

8.2.5 New Models of Memory

Section 4.5 discusses the problems and limitations relating to the memory model and reasoning engine used by REDMAP to access knowledge. REDMAP, and DMAP systems at large, use memory as a source of remindings and expectations and do not, in general, perform a lot of deep reasoning beyond leveraging generalization hierarchies. The algorithms presented in this dissertation would benefit from better support for several commonly needed memory operations. Currently, when multiple episodes match what REDMAP is reading about, remindings are retrieved exhaustively and iterated over one at a time. Models of memory that could allow REDMAP to specify a set of properties for an episode and know just if such a memory exists or not, or possibly a count if there are only a few that match otherwise “many,” would allow all similar episodes to be processed as a group. It would further accelerate processes relating to remindings if features could be incrementally added in order to update a set, or if two sets could be intersected efficiently. An additional desirable property of a reminding-oriented memory would be the ability to retrieve these sets such that they include episodes that are consistent with the query not just confirmed (see section 4.2.1 for discussion of confirmed and consistent).
Support for refinement, and easy access to generalizations to build expectations would also be extremely useful.

8.3 Final Remarks

Most domain-independent models of memory are not optimized for the interactions required by REDMAP. Some, such as those used by REDMAP, can reason over a broad range of domains, and are represented using deeply structured ontologies and semantics. They can combine pieces in novel configurations but are typically optimized for reasoning over reminding. Other domain-independent memories, such as most CBR systems, are optimized for reminding. However, they are typically organized with domain-dependent knowledge representations, and are not integrated with other memories or structured ontologies. Next-generation models of semantic and episodic memory and knowledge representation are needed to support broad, open domain reminding and learning with deeply structured memory. These models are needed in order to facilitate tasks like learning by reading, or even more narrowly, the kind of reference resolution presented in this dissertation. We are at a time where large quantities of knowledge are being formally represented, computational power is sufficient to manipulate it, and there is strong interest in tasks that require it. It is an ideal time for exploring both new models of episodic memory and language understanding, tasks that require these resources. I believe breakthroughs in one will go a long way to facilitate the other.

Finally, although this dissertation is focused on language understanding, the ideas I put forth are primarily about resolving references to episodic memory. References in this dissertation
happened to be generated by text via reading, however I believe these ideas can be more broadly applied to any reference encountered by the machine. References may be *linguistic* such as those discussed in this dissertation or those produced in dialog, *perceptual* including visual references to objects in an environment, and even purely *cognitive* references such as those that may be produced by analogy or speculation. All of these types of understanding would be fundamentally improved by better models of reminding and memory and algorithms for reference resolution.
References


Appendix A

This appendix contains a sample of the simplified English texts in the evaluation corpus (see chapter 2) used in the REDMAP experiments. The first and fourth texts for each of the nine sections of the extended corpus described in table 2.2 are provided.

Election

election-afgan20041008cnn-1-1-1

An election is occurring in Afghanistan.
The election started on October 8, 2004.

election-afgan20041008cnn-4-1-4

An election is occurring in Afghanistan.
There are 16 candidates competing in the election.
President Hamid Karzai is a candidate.
Hamid Karzai is the favorite in the election.
Abdul Rashid Dostum is a candidate.
Massouda Jalal is a woman.
Massouda Jalal is a candidate.

Geography

graphy-egypt-1-1-3

Egypt is in northeast Africa.
Egypt is at the northeast corner of Africa.
Egypt shares a border with the Mediterranean Sea.
Egypt borders Libya on the west.
Egypt borders Sudan on the south.
Egypt borders the Red Sea on the east.
Egypt borders Israel on the east.
Egypt is nearly one and one-half times the size of Texas.
There are regions in Egypt.
The two regions are extremely arid.
The two regions have different sizes.
The Nile River divides the two regions.
The Nile River traverses Egypt.
The Nile River is the dominant feature of the landscape of Egypt.
The Nile River flows northward.
The Nile River starts 100 mi south of the Mediterranean Sea.
The Nile River starts 161 km south of the Mediterranean Sea.
The Nile River fans out to a sea front.
The sea front is 155 mi long.
The sea front is between Alexandria and Port Said.

geography-israel-1-1-6

Israel is slightly larger than Massachusetts.
Israel shares a border with the Mediterranean Sea.
Israel borders Egypt on the west.
Israel borders Syria on the east.
Israel borders Jordan on the east.
Israel borders Lebanon on the north.
There is a maritime plain in Israel.
The plain is extremely fertile.
The Negev region is in southern Israel.
The Negev region comprises half of the total area of Israel.
The Jordon River is in Israel.
The Jordon River is the only important river in Israel.
The Jordon River starts in the north of Israel.
The Waters of Merom are in Israel.
The Jordon River traverses Lake Hule.
The Sea of Galilee is in Israel.
The Jordon River traverses the Sea of Tiberias.
The Jordon River enters the Dead Sea.
The Dead Sea is 1349 ft below sea level.
The Dead Sea is 411 m below sea level.
The Dead Sea is at the world's lowest land elevation.
History

*history-egypt-01-1-3*

Egyptian history dates back to about 4000 B.C.
The kingdom of upper Egypt and the kingdom of lower Egypt were united.
The unity existed in 4000 B.C.
The kingdom of upper Egypt was highly sophisticated in 4000 B.C.
The kingdom of lower Egypt was highly sophisticated in 4000 B.C.

*history-egypt-04-1-2*

Alexander the Great conquered Egypt.
The occupation started in 332 B.C.

Lebanon

*conflict-lebanon-01-1-1-01*

Hezbollah attacked Israel.
The attack killed 3 Israeli soldiers.
Hezbollah kidnapped 2 Israeli soldiers.
The attack occurred on July 12, 2006.
The kidnapping occurred on July 12, 2006.
Israel bombed Lebanon more than 100 times.
The bombing was in response to the kidnapping.

*conflict-lebanon-03-1-1-02*

Israel bombed the highway between Beirut and Damascus.
The bombing occurred on July 14, 2006.
Military

*militaryaction-14marines20050803-1-1-1*

An attack occurred in Al Anbar.
The bombing occurred on August 3, 2005.
The attack killed 14 soldiers.

*militaryaction-offensive20050810fox-1-1-2*

An offensive occurred in the Euphrates River valley.
The Euphrates River valley is in western Iraq.
The purpose of the offensive is to disrupt insurgents.
The insurgents are in the Euphrates River valley.
The offensive was performed by the U.S. military.
The offensive involves about 1000 U.S. Marines and Iraqi soldiers.

Organization

*old-organization-eta-1-1-14*

ETA was founded in 1959.
ETA has a goal.
The goal is to establish a homeland.
The homeland would be independent.
The homeland would be based on Marxist principles.
The homeland would encompass Vizcaya.
The homeland would encompass Guipuzcoa.
The homeland would encompass Alava.
The homeland would encompass Navarra.
The homeland would encompass Labourd.
The homeland would encompass Basse-Navarra.
The homeland would encompass Soule.
organization-aljihad-3-1-1

Al Jihad was active since the 1970s.
Al Jihad has a primary goal.
The goal is to overthrow the Egyptian Government.
The goal is to establish an Islamic state in Egypt.

Person

person-binladen-1-1-1

Osama bin Laden was born in 1957.
Osama bin Laden came from a wealthy family.
Osama bin Laden came from a Saudi Arabian family.
The family owns a business.
The business does construction.
The business is multinational.
Osama bin Laden was trained as an engineer.
Osama bin Laden inherited wealth.

person-faisal2-2-1-3

The father of Faisal was King Ghazi.
The father of Faisal II died.
Faisal II became king after the death.

Relation

relations-egypt-05-1-1

There was a treaty in 1899.
The treaty created a boundary.
The boundary is along the 22nd Parallel.
There are triangular areas.
The areas extend north of the boundary.
The areas extend south of the boundary.
Egypt had a claim to administer the areas.  
Egypt retains that claim.  
Egypt has withdrawn their military presence from the areas.  
Sudan also had a claim to administer the areas.  
Sudan retains that claim.  
Sudan has withdrawn their military presence from the areas.

relations-iran-04-1-1

Iran has disputes with the UAE.  
The disputes are about Iran's occupation of the Tunb Islands.  
The disputes are about Iran's occupation of Abu Musa Island.  
UAE has engaged in direct talks with Iran to resolve the disputes.  
The Arab League supports the resolution of the disputes.

Terrorism

terroristattack-aqaba20050819-1-1-2

An attack occurred on August 19, 2005.  
The attack occurred in Aqaba.  
Rockets were fired at U.S. Navy ships.  
The rockets missed the ships.  
The ships were in Aqaba.

terroristattack-aqaba20050819-1-4-1

An attack occurred on August 19, 2005.  
The attack occurred in Aqaba.  
The attack wounded 1 Jordanian.  
Abdullah al-Azzam Brigades claimed responsibility for the attack.  
Abdullah al-Azzam Brigades is associated with al-Qaeda.  
Abdullah al-Azzam Brigades is in Egypt.  
The claim could not be authenticated.