INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

SEMI-AUTOMATIC RECOGNITION OF SEMANTIC RELATIONSHIPS IN ENGLISH TECHNICAL TEXTS

Dissertation
submitted to the School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

June, 1998

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

ABSTRACT

When people read a text, they rely on *a priori* knowledge of language, common sense knowledge and knowledge of the domain. Many natural language processing systems implement this human model of language understanding, and therefore are heavily knowledge-dependent. Such systems assume the availability of large amounts of background knowledge coded in advance in a specialized formalism. The problem with such an assumption is that building a knowledge base with sufficient and relevant content is labour-intensive and very costly. And often, the resulting knowledge is either too specific to be used for more than one very narrow domain or too general to allow subtle analyses of texts.

In order to avoid the problems of manually encoding background knowledge, many researchers have abandoned symbolic language analysis in favour of statistical methods. The availability of large online corpora and improvements in computing resources have made it possible to make predictions about meaning based on observations of frequencies, contexts, correlation, and other phenomena in a corpus. Systems that use statistical methods have had some impressive successes, notably in part of speech tagging, word class clustering and word sense disambiguation. But these systems often require large amounts of analyzed language data to arrive at even shallow interpretations.

Both of these kinds of natural language processing systems seek models of a text—knowledge-intensive systems a deep semantic model, corpus-based systems a much shallower distributional one. And both kinds of system depend on outside sources of data. This dissertation describes the construction and evaluation of an interactive tool that also seeks a model of a text. The model takes the form of semantic relationships between syntactic elements in English sentences. The system also depends on an outside source of data: a cooperative user. Unlike knowledge-intensive and corpus-based systems, however, it does not require a large repository of semantic information and it does not require any

previously analyzed data: it can start processing a text from scratch. The system inspects the surface syntax of a sentence to make informed decisions about its possible interpretations. It then suggests these interpretations to the user. As more text is analyzed, the system learns from previous analyses to make better decisions, reducing its reliance on the user. Evaluation confirms that the semi-automatic acquisition of the model of a text is relatively painless for the user.

The regular structure of the model identifies concepts that have different surface-syntactic forms. These concepts could be used as the knowledge base for expert systems or query answering systems. They could be used as a conceptual profile of a text, allowing, for example, text indexing on semantic concepts instead of just keywords. The concepts and semantic relationships between them could serve as base structures for text summarization. They could also be used as the domain-specific background knowledge core for natural language processing systems that attempt deeper understanding of a text.

ISTHMI

Every man is a peninsula.

I would like to acknowledge here all the people who contributed in some way to the development of this dissertation. But I can't. There isn't nearly enough space. Those who stopped to talk to me about my research when I needed inspiration, those who stopped talking to me about my research when I needed ventilation, those who shared sounds or words or ice time, those who with me supported my local brewery can be assured that I appreciate them, every one.

Then there are those whose contribution cannot be dismissed with cute coordination. Stan Szpakowicz gave me guidance in every facet of my academic life. His far-sighted direction is what led me here, and I'm grateful.

I would like to acknowledge my thesis committee: Jean-Pierre Corriveau, Stan Matwin, Ingrid Meyer and Fred Popowich each read and understood the dissertation thoroughly, attacking it on different levels from different angles. Their probing gave me a better understanding of my own thesis.

The "TANKA core" is my immediate research family. Sylvain Delisle's dedication helped make my work workable, and then he made it fun. Terry Copeck's talent and attention to a million things kept TANKA running, in spite of obstacles both man-made and natural.

Sylvia Boyd, Rob Holte and Stan Matwin deserve my gratitude for all the times I saw them go out of their way to open doors for me—and more for all the times I didn't.

My parents, Bob and Betty Jo, supported me in the ways that parents usually get acknowledgment for. That they supported me as an equal was unique. They have earned my thanks and admiration, and have shown me a friendship that rarely transcends generations. My brothers Thomas and Stephen and my sister Louise provided

entertainment, sport, humour, warmth and really good food. If you could choose your family, you would choose mine.

Johanne Morin. She knows my algorithms, she knows my heart, she knows I don't like to shoot sides, she knows how to make me happy. She is beauty, intelligence, taste and mirth. If she could play nets, she would be perfect.

This research was supported by the Natural Science and Engineering Research Council of Canada.

CONTENTS—GENERAL

CONTENTS—DETAILED

FIGURES

TABLES

1 Overview of the Project

There is no *one-to-one* mapping from syntactic relationships to semantic relationships in English sentences. A sentence with a main clause and a subordinate clause does not always express a causal relationship between two events. The subject of a verb does not always represent the instigator of an act. An adjective does not always indicate a physical property of some entity. Yet there is *a* mapping from syntactic relationships to semantic relationships; some part of the meaning of a sentence is reflected in its surface-syntactic form.

This dissertation is about finding links between syntactic and semantic relationships. In particular it is about defining sets of semantic relationships that can be expressed by syntactically related sentence elements. It is about cataloguing clues that hint at the syntax-to-semantics mapping and using those clues in a program to uncover semantic relationships where they exist in sentences. It is also about recognizing when the clues are insufficient. When this occurs, linguistic facts lurking beneath the surface become

necessary; they can be learned from user assistance to make uncovering subsequent semantic relationships easier and more accurate.

1.1 Automated Semantic Analysis

Automated semantic analysis of texts involves such processes as determining word senses and referents, recognizing modification relations and semantic roles, interpreting quantification, and many others. Whatever the processes involved, automated semantic analysis usually attempts to uncover components of the meaning of a fragment of text beyond its surface-syntactic structure. Consider sentences (1) and (2).

(1) *Ed broke the thumb on Joe's left hand when he swung the hammer off-target.*

(2) *Ed swung a heavy metal hammer askew and broke Joe's left hand thumb.*

There are many syntactic and lexical differences between (1) and (2). Yet the events and participating entities are similar enough that if both sentences occurred together in a text, one of them might be considered redundant. Semantic analysis should produce structures that reveal some of the underlying similarities of meaning in the sentences.

1.1.1 Oracles

Automated semantic analysis does not necessarily imply full automation or autonomy. Often some outside source of knowledge is required to inform analysis. This oracle may be a large, comprehensive knowledge base, a corpus of analyzed text or a human user—as in this project.

1.1.2 Entities and Events

Since this dissertation deals with semantic relationships marked by syntactic forms, at times I will find it necessary to talk about the semantic *things* that syntactic elements refer to.

Frawley (1993: xiv) identifies certain "basic objects in a model of the world: things, actions, and the relation of things to actions." *Things* more formally are *entities*: "relatively stable and atemporal discourse, ontological, and conceptual phenomena" (p. 68) and they usually surface as nouns. *Actions* more formally are *events*: "relatively temporal relation[s] in conceptual space" (p. 144) and they usually surface as verbs.[1] Finally, the relation of things to actions is accounted for by *thematic roles* (cases): "grammatically relevant relations between predicates (often events) and arguments (often entities)" (p. 199).

Borrowing liberally from Frawley, then, I will use the term *entity* to refer to the semantic thing represented in syntax as a noun. I will also refer to the semantically complex thing represented in syntax by a noun phrase as an *entity*. Modifiers of a noun in a noun phrase serve to ground it, but the complex noun phrase still refers to a single entity. Using this terminology, both (3) and (4) are entities.

(3) *cat*

(4) *the fat cat on the porch*

Similarly, I will use the term *event* to refer to the semantic thing represented in syntax as a verb. I will also refer to the semantically complex thing represented in syntax by a clause as an *event*. Both (5) and (6) are events.

(5) *snooze*

(6) *the fat cat on the porch snoozes all day long*

1.2 Goals of the Project

The semantic relationships in the sentences of a text form a model of that text. With the present state of the art, fully automatic construction of such a model is not feasible

[1] Frawley's definition of event is general enough to include four more specific kinds: *acts*, *states*, *causes*, and *motion*.

without a knowledge base. A fully manual construction of the model is feasible, but onerous. My thesis is that it is possible to acquire the model interactively without demanding of a user as much as a large knowledge engineering effort. This claim can be tested through the following specific project goals. The extent to which the goals have been met will be investigated in chapter 6.

1 *To construct usable, portable sets of semantic relationships.* The sets should take into account similar sets proposed by others and should be general enough for any technical text and for other kinds of relationship-based analysis of texts. Construction should be informed by empirical considerations, such as marker data and text coverage.

2 *To design and implement a semi-automatic system for recognizing semantic relationships in text.* The system should make use of whatever grammatical knowledge is available, but should avoid open-ended, domain-specific semantic knowledge. It should also offer informed suggestions and learn from user input.

3 *To evaluate the ability of the system to learn.* As more text is processed, the number of semantic relationships recognized automatically should increase, reducing reliance on the user.

4 *To evaluate the system's coverage of relationships in a text.* Parse tree input is not always perfect. The system should be able to recognize some semantic relationships even from imperfect parses.

5 *To evaluate the burden the system places on the user.* Recognizing semantic relationships is not always easy, even for people. The system should be able to analyze reasonable amounts of text in a reasonable amount of time without overtaxing the user.

1.3 Semantic Analysis in TANKA

TANKA (Text ANalysis for Knowledge Acquisition) is a project whose goal is a system that produces a model of the knowledge contained in a technical text. Syntactic analysis in TANKA is performed by DIPETT; semantic analysis is performed by HAIKU. The design, implementation and evaluation of HAIKU are the vehicle for accomplishing the goals listed in the previous section.

1.3.1 DIPETT Parse Trees

DIPETT (Domain Independent Parser of English Technical Texts) is a broad-coverage Definite Clause Grammar parser (Delisle 1994, Delisle & Szpakowicz 1995) whose rules are based primarily on Quirk *et al.* (1985). DIPETT automatically produces a single initial parse. This first good parse tree is a detailed representation of the constituent structure of a sentence. In particular, the parse tree contains the noun phrases, clauses and syntactic connections between clauses that make up the input for HAIKU semantic analysis.

1.3.2 Three-Tiered Semantic Analysis

Semantic analysis in TANKA is the semi-automatic recognition of semantic relationships among entities and events represented in sentences. HAIKU recognizes three kinds of semantic relationships: clause level relationships, cases and noun modifier relationships. This division of semantic relationships is motivated by their syntactic manifestation at different levels. It is possible, however, for the same semantic relationship to be realized at any of the levels, suggesting that a unified view of semantic relationships is appropriate. For this project the relationships that apply at each level remain separate, and I investigate the possibility of unifying them in section 5.4.

At each level HAIKU attempts to find the best relationships using syntactic evidence and previous analyses. The system suggests these relationships to a cooperating user for approval. As more sentences are analyzed, HAIKU learns to make better suggestions based on the user's input.

Clause Level Relationships

Clause level relationships (CLRs) are semantic relationships between events represented syntactically by syndetically connected finite clauses. The CLR analyzer recognizes these relationships in sentences by considering the lexical items that mark them (coordinators, correlatives, subordinators) and the syntactic features of the connected clauses.

Case Relationships

Cases are semantic relationships that express the roles of the participants and the circumstances of an event. The case analyzer recognizes cases in finite clauses by considering the syntactic or lexical items that indicate verb arguments (subject, objects, prepositional phrases, adverbials).

Noun Modifier Relationships

Noun modifier relationships (NMRs) are semantic relationships that define entities. They are expressed within noun phrases as modification. The NMR analyzer recognizes relationships expressed by adjectives, premodifying nouns and postmodifying prepositional phrases and appositives.

1.3.3 Background Knowledge

HAIKU can be characterized as knowledge-scant. It is designed to avoid excessive reliance on precoded, domain-dependent, semantic knowledge. In practice certain kinds of knowledge encoding are forbidden in TANKA and certain kinds are allowed. The distinction is not entirely arbitrary. In this section, I will describe what knowledge *is* allowed in TANKA, and how it differs from the knowledge that is not.

Linguistic Knowledge

Both DIPETT and HAIKU use all kinds of linguistic knowledge. DIPETT's grammar is linguistic knowledge. The fact that part of the meaning of an utterance is reflected explicitly in its surface structure (*e.g.*, that an entity is represented syntactically as a noun and an event as a verb) is linguistic knowledge. That modal auxiliaries express uncertainty, that prepositional and adverbial phrases indicate circumstances of an event,

that certain kinds of noun compounds are hyponyms of their heads are all bits of linguistic knowledge, and are all allowed by TANKA.

These examples are knowledge of language in general. Such knowledge is closely tied to the surface syntax and does not vary from domain to domain or from text to text. It is independent in the sense that each linguistic fact can be encoded to assist the semantic analysis task without requiring that further knowledge (such as lexical semantics) be added for the fact to be useful.[2] Furthermore, the linguistic knowledge in TANKA is relatively uncontroversial. There may be disagreement on the specific semantic relationships possible between adjectives and the nouns they modify, but there is relatively less disagreement on the fact that such relationships exist.

Semantic Knowledge

The sets of semantic relationships recognized by HAIKU are semantic knowledge (sections 2.3, 3.4 and 4.4). The dictionaries mapping lexical markers to subsets of relationships are semantic knowledge (sections 2.4, 3.3.1 and 4.5). The preference rules for clause level relationships and the paraphrases for noun modifier relationships are semantic knowledge (sections 2.5.3 and 4.4.1).

The semantic knowledge allowed in TANKA must be closed. Mapping markers to semantic relationships is allowed since the sets of markers and relationships are small, fixed and domain-independent: they do not grow or change from domain to domain. Tagging nouns with semantic class information is not allowed in TANKA, since the nouns are an open category and their senses may change from domain to domain. The number and nature of preference rules for CLRs depend on the CLRs, not on the semantics of the main verbs in clauses. For example, it might be tempting to add a rule stating that a clause with the main verb *prevent* is evidence for the Prevention CLR (section 2.3.1), but such rules would be potentially open-ended. Case assignment would be relatively simple if the case

---

[2] An example of linguistic/semantic knowledge that does not have this property is the knowledge that nouns denoting animate objects are more likely instigators of acts. That knowledge would require that nouns in a lexicon be tagged as `±animate`.

patterns associated with particular verbs were known in advance. Again, that knowledge is open-ended, and not allowed in TANKA.

The techniques used by HAIKU to recognize semantic relationships should be independent of the allowed semantic knowledge. The noun modifier relationship analyzer can work with any set of noun modifier relationships, and even allows run-time addition of new NMRs (section 4.6.5). CLR analysis uses the preference rules for pairs of CLRs if such rules exist. If not, the CLRs are given equal weight in CLR competitions (section 2.5.4), meaning that HAIKU could assign CLRs even with no prior knowledge of CLR semantics. In the absence of marker dictionaries, HAIKU would be more dependent on user input, but could still improve over time by learning from user assignments.

User Knowledge

In the absence of other kinds of semantic knowledge, HAIKU relies on the user to supply information. It is therefore a priority in HAIKU to minimize the burden placed on the user in interactions with the system. Evaluating the decrease in user activity over the course of analyzing a text measures HAIKU's success in curbing user burden.

1.4 Evaluation

A significant part of this dissertation is devoted to the evaluation of HAIKU's performance on various texts and of the knowledge it acquires. The evaluation experiments were necessarily specific to TANKA, but I have also paid attention to some more general evaluation principles.

There has been increased interest in the evaluation of natural language processing (NLP) tools over the last few years. The message understanding conferences (MUCs) are a direct reaction to a lack of standardization in the evaluation of such tools. The MUC competitions compare the performance of different systems on some uniform, predetermined text processing task. The motivation for and background of the MUC competitions is described in Grishman & Sundheim (1996).

Although these competitions have successfully emphasized the importance of evaluation for the NLP community, they have also revealed certain limitations. In particular, researchers have noted that the predefined tasks in evaluation competitions result in applications that are designed to score well, but are not portable (see MUC-6 1996). Furthermore, little attention has been paid to the evaluation of interactive systems and the role of users (Hirschman & Thompson 1996).

To address these issues, Sparck Jones (1994) and Sparck Jones & Galliers (1996) offer more general strategies for evaluating generic NLP systems: "there is far too much variety in the situations and subjects of evaluation to come up with a definite [evaluation] scenario" (Sparck Jones & Galliers 1996: 193).

Hirschman & Thompson (1996) distinguish two kinds of evaluations—*diagnostic evaluations* and *performance evaluations*—that make different assumptions about the distribution of test data. A diagnostic evaluation tests a system on all of the linguistic phenomena that it is designed to handle. Lehmann *et al.* (1996) describe the construction of an exhaustive test suite for diagnostic evaluations of natural language processing systems. The distribution of linguistic phenomena in the test suite of a diagnostic evaluation is almost certainly not the same as the distribution of phenomena in complete texts, which form the test suite for a performance evaluation. For performance evaluations it is important to identify *criteria* (what is being evaluated), *measures* (what properties of performance reflect the criteria) and *methods* (how the measures are analyzed to arrive at an evaluation of the criteria).

The components of HAIKU will be evaluated on four general criteria:

1 the number of semantic relationships recognized correctly by the system;

2 the increase in proportion of correct analyses over the course of a session (HAIKU's ability to learn);

3 the system's coverage of the test texts (number of semantic relationships recognized relative to the number actually in the text);

4 the burden that interaction places on the user.

The first two criteria are objective and can be measured directly. Coverage is objective, but cannot always be measured directly, since the number of semantic relationships in an entire text is not always easy to determine. I will use sampling when direct measurement is not feasible (see sections 3.6.1 and 4.7.2). The fourth criterion is mostly subjective, but its evaluation will take into account quantitative measurements of user participation (see sections 1.4.3 and 1.4.4).

1.4.1 Test Texts

TANKA targets unedited English technical text for semi-automatic analysis. Technical texts usually lack humour, intentional ambiguity, and other devices that would make analysis more difficult. Copeck *et al.* (1997) describe a study to investigate the character of texts typically considered technical.

DIPETT and HAIKU have been tested on a variety of technical texts. I refer to four of these experiments repeatedly throughout the dissertation.

The *clouds* experiment was a performance evaluation summarized in Barker & Delisle (1996). The text for the experiment was the *Junior Science Book of Rain, Hail, Sleet & Snow* (Larrick 1961). The *clouds* text has fairly simple syntax and was chosen to ensure a high number of correct parses. The belief was that the higher parse success rate would allow the system to recognize more semantic relationships, since HAIKU would have access to more correct parse trees. Unfortunately, the simpler text uses very general terminology and less complex modification, resulting in fewer semantic relationships. The *clouds* experiment was meant to evaluate DIPETT, CLR analysis and case analysis as well as user burden. Sylvain Delisle and I were the two users in the experiment.

Results of the *small engines* experiment appear in Barker *et al.* (1998). *The Mechanics of Small Engines* (Atkinson 1990) has more complex syntax, resulting in fewer correct parse trees available for HAIKU. On the other hand, the text contains more semantic relationships per sentence. This experiment was a performance evaluation of DIPETT, CLR analysis, case analysis, NMR analysis and user burden. Sylvain Delisle and I were again the users in the experiment.

The *building code* experiment used sentences from the Ontario Building Code (Ontario Ministry of Housing 1991) in a diagnostic evaluation of the CLR analyzer. The *building code* text contains thousands of complex sentences with a variety of conjunctions connecting clauses and a variety of syntactic verb phrase features.

The *sparc* experiment was a performance evaluation of components of the NMR analyzer on noun phrases from the *SPARCstation 1 Installation Guide* (Chan 1989).

1.4.2 Parser Evaluation

For the *clouds* and *small engines* experiments, the quality of DIPETT's output had a major effect on semantic analysis.

The *clouds* experiment applied DIPETT and HAIKU to 512 sentences. DIPETT produced a single complete parse tree covering all tokens (words and punctuation) for 412 sentences (80%). A complete parse does not necessarily mean a correct parse. 47% of the parses of *clouds* sentences were perfect. Another 19% were good enough not to affect semantic analysis. The other 34% of the parse trees had errors that were serious enough to prevent the recognition of one or more semantic relationships by HAIKU.

In the *small engines* experiment, 55% of the 557 sentences received a complete parse. Due to the much more complex syntax, only 31% of the parses were perfect and another 9% were good enough not to affect semantic analysis. That means that for 60% of the sentences in the *small engines* text, parse errors were serious enough to prevent HAIKU from recognizing all the semantic relationships.

The individual evaluations of CLR analysis (section 2.6), case analysis (section 3.6) and NMR analysis (section 4.7) investigate the extent to which HAIKU was affected by parse errors.

1.4.3 User Actions

The evaluation of HAIKU's success at recognizing semantic relationships will be judged by the type of action required by the user. Every time HAIKU assigns a semantic

relationship to an input, it requires approval from the user, so even correct analyses involve user interaction.

The first type of user action is *accept*: HAIKU presents the user with one single semantic relationship suggestion and the suggestion is correct. *Accept* actions are considered correct semantic relationship assignments by the system.

The second type of action is *choose*: HAIKU suggests more than one possible semantic relationship and the correct relationship is among them. The number of suggestions is limited to the number of marker→relationship mappings for a given marker, which is at most roughly half the total number of semantic relationships. In practise the number of suggestions is usually no more than three or four (see for example sections 3.6.1 and 4.7.2).

The third action is *supply*: either HAIKU is unable to offer any suggestions, or the correct relationship is not among HAIKU's suggestions; the user must supply the relationship.

In certain instances it will be convenient to compare the number of *supply* actions to the sum of the other two types. In those comparisons I will use the term *system assignments* to refer to semantic relationships assigned as a result of a *choose* or *accept* action and *user assignments* to refer to relationships assigned as a result of a *supply* action.

1.4.4 User Burden

For each user interaction in the *clouds* and *small engines* experiments, the degree of difficulty of interaction (referred to as *onus*) was recorded as an integer from 0 to 3. 0 means that the interaction is trivial. 1 is assigned to an interaction that requires a few moments of reflection. 2 rates an interaction as requiring serious thought, but eventually a semantic relationship is assigned. 3 means that even after much contemplation no relationship is deemed appropriate for the given input.

User burden is also reflected in the amount of time required to oversee the analysis of a text. The average number of tokens (words or punctuation) in the *small engines* sentences was 15.4. The average number of CLRs per sentence was 0.04, while the average number

of case patterns was 1.05 and the average number of NMRs was 1.59. On average, we spent 1 minute, 49 seconds on each sentence. By coincidence, the average user time for the *clouds* experiment was also 1 minute, 49 seconds for an average of 0.10 CLRs and 0.86 case patterns per sentence (NMRs were not evaluated in the *clouds* experiment).

1.5 Applications

Information extraction from text often involves the identification of essential participants and circumstances in acts. The participants and circumstances can be represented directly as cases. More specific knowledge of the participants involves recovery of the relationships within noun phrases (NMRs).

Template filling is a kind of information extraction. A template has a fixed set of slots to be filled with specific information from the text. For events, these slots are often analogous to cases (with names such as Agent, Instrument, Location, etc.) and sometimes also relate events to other events (comparable to recognizing CLRs). For entities the slots often correspond to hypernyms, subcomponents, purposes and properties—knowledge of the kind that is captured by NMR analysis.

It is often claimed that case is a universal phenomenon across natural languages (Fillmore 1968). Research has identified strikingly similar lists of cases for many human languages (see Campe 1994 for a multilingual bibliography of case). The semantic roles in a source language clause could be used as part of an *interlingua in machine translation.*

Question answering systems are often faced with questions about properties of entities and the participants and circumstances of events.

The construction of *semantic lexicons* has become a popular area of research in natural language engineering. Noun entries in these lexicons often identify properties of entities, hypernyms, subcomponents, as well as the events in which the entity expressed by the noun participates.

1.6 Organization of the Dissertation

The five goals of section 1.2 apply to all three levels of processing in HAIKU. This dissertation is divided into three main chapters corresponding to HAIKU's clause level relationship analysis, case analysis and noun modifier relationship analysis. Each chapter describes work done towards satisfying the project goals: building a set of semantic relationships; implementing a semi-automatic system that uses linguistic clues to recognize semantic relationships in sentences; evaluating the system on its ability to learn, the coverage of its relationships and the burden it places on the user.

Clause level relationship analysis, since it deals with the largest (and most intricate) parse tree fragments, has access to more linguistic evidence for recognizing relationships. The connection between syntax and semantics is stressed more in chapter 2 than in the other chapters, simply because there is more syntax at that level than the others.

Case analysis is not part of the contribution of this project, since the techniques have already been described in Delisle (1994) and Delisle *et al.* (1996). What *is* part of this project is the process of constructing the set of cases and the evaluation of their coverage. HAIKU's cases have received more attention in evaluation than either CLRs or NMRs. Chapter 3 is primarily about defining and evaluating a set of semantic relationships.

There is little surface-linguistic evidence to guide the assignment of noun modifier relationships. The NMRs themselves have not been tested for coverage to the extent that the cases have been. What is unique about the NMR analyzer is its use of partial matching on a growing base of previous instances, and the attention paid to elements of the user interface. Chapter 4 has an emphasis on learning and user interaction that does not occupy as prominent a place in the other chapters.

Chapter 5 explores reasonable extensions to the project along with bolder departures. Chapter 6 summarizes the project and evidence that its goals have been met. Three appendices contain samples of the clause level relationship marker dictionary (see section 2.4), the case marker dictionary (section 3.3.1) and the noun modifier relationship marker

dictionary (4.5). Literature review has been parcelled out to sections 2.1, 3.1 and 4.1. There is an abundance of interesting work related indirectly to each of the three main areas of the project, but less that is directly applicable.

1.6.1 Paper Map

Parts of the dissertation have already been the subject of technical reports and papers:

topic	*dissertation sections*	*previously described in*
CLR Analysis	2.1-2.4, 2.5.1-2.5.4, 2.6.1, 2.6.2	Barker (1994), Barker & Delisle (1996), Barker & Szpakowicz (1995), Barker *et al.* (1998)
Cases	3.1-3.4, 3.6.2	Barker (1996), Barker & Delisle (1996), Barker *et al.* (1997)
Case Analysis	3.5, 3.6.1	Barker & Delisle (1996), Barker *et al.* (1998), Delisle *et al.* (1993), Delisle *et al.* (1996),
NMR Analysis	4.1, 4.2, 4.4-4.6, 4.7.2	Barker (1997), Barker & Szpakowicz (1998), Barker *et al.* (1998)
Bracketing	4.3, 4.7.1	Barker (1997), Barker (1998), Barker *et al.* (1998)
Evaluation	2.6.1, 2.6.2, 3.6, 4.7	Barker & Delisle (1996), Barker *et al.* (1997), Barker *et al.* (1998)

2 Clause Level Relationship Analysis

Clause level relationships are semantic relationships between clauses within a complex sentence. Syntactic analysis identifies the type of syntactic connection between the clauses, the lexical connector and syntactic features of the main verb phrase in each clause. The clause level relationship analyzer chooses a relationship among candidates competing on the basis of syntactic verb features and the semantics of the relationships.

This chapter is about clause level relationship analysis (CLRA), the first stage of analysis in HAIKU. The parse trees that are input to CLRA are more intricate than those for either case analysis or noun modifier relationship analysis. The emphasis is on using syntactic evidence to guide the assignment of clause level relationships (CLRs).

2.1 Introduction

This introduction gives background on semantic relations between clauses, research in discourse analysis and research on the semantics of verb sequence features relevant to CLRA. In section 2.2 I describe the kinds of syntactic information provided by DIPETT and used in assigning CLRs. Definitions and examples of the CLRs appear in section 2.3. Section 2.4 describes the construction of the dictionary of mappings from conjunctions and adverbs to CLRs. Section 2.5 contains details of the process of assigning CLRs to clauses; the role of verb sequence features, CLR semantics and user input are discussed in detail. The results of two different kinds of evaluation are given in section 2.6: a more controlled test of CLRA on the range of possible inputs, and tests on texts where syntactic phenomena are less uniformly distributed. Section 2.7 shows an example of typical CLRA interactions.

2.1.1 Semantic Relationships

One of the earlier attempts in computational linguistics to enumerate a set of semantic relationships between propositions was an extension to Schank's Conceptual Dependency Theory (Schank 1975). Although the original theory only defined primitive acts and the relationships between acts and their participants, a later refinement to the theory (Schank & Abelson 1977) introduced Conceptual Relations (including Enable, Result, Reason, and Initiate) to capture the semantic relationships between acts.

Halliday & Hasan (1976) present several Conjunctive Relations divided into two *planes* of relations: External and Internal. The Conjunctive Relations are further divided into Additive, Adversative, Causal and Temporal relations. There is a close correspondence of the External relations within these categories to HAIKU's Conjunctive, negative Causal, positive Causal and Temporal CLRs (see section 2.3).

Research on relationships between clauses (and larger text units) sometimes distinguishes between semantic relations and pragmatic relations. Semantic relations hold between the events represented by clauses; pragmatic relations deal with the communicative function

of the clauses themselves. Van Dijk (1977) gives a thorough treatment of semantic versus pragmatic considerations in the study of discourse and presents sets of different kinds of Connections. The kind of Connections relevant to HAIKU's CLRs are van Dijk's Semantic Connections, which include Conjunction, Disjunction, Conditionals (including causal relations) and Contrastives.

Hobbs (1983) presents a taxonomy of Coherence Relations that includes Enablement, Cause and Contrast (similar to HAIKU's Detraction). Since Hobbs was interested in coherence of discourse, the taxonomy also includes several pragmatic relations.

Bäcklund (1984) identifies six types of clauses: Temporal, Concessive, Conditional, Comparative, Conditional-Concessive and Locative. These clause types are defined by their semantic function within the discourse. For each type, she enumerates a set of connectives that typically introduce clauses of that type. For example, Temporal clauses are introduced by the connectives *when*, *whenever*, *after*, *as long as*, etc. All clauses introduced by temporal connectives are lumped together as Temporal functions. Since not all clauses introduced by the so-called temporal connectives express a temporal relationship, Bäcklund further divides the Temporal functions into temporal and non-temporal relations.

Mann & Thompson's Rhetorical Structure Theory (RST) offers a list of Rhetorical Relations that includes relations roughly corresponding to all of HAIKU's CLRs (Mann & Thompson 1986a, 1986b, 1988). As with Hobbs' Coherence Relations, however, Mann & Thompson's list is aimed at capturing both semantic and pragmatic relations. Several authors have proposed sets of relations based on RST (including Hovy 1993, Lascarides *et al.* 1992 and Sanders *et al.* 1992).

Schiffrin (1987) presents a list of Discourse Relations based on a study of the lexical items (discourse markers) that signal them. Although many of the markers are unique to conversational speech (such as *oh*, *well*, *y'know*, etc.), the resulting set of relations has much in common with HAIKU's CLRs, including a division into three categories for Conjunctive, Causal and Temporal relations.

CONJUNCTIVE	
Conjunctive	Coordinative
Continuative	Contrastive
Disjunctive	

CAUSAL	
Cause-Result	Warrant-Inference
Motive-Action	

TEMPORAL	
Reference Time	Event Time
Discourse Time	

Table 1: Discourse relations from Schiffrin (1987)

Dahlgren (1988) summarizes the work of a number of researchers (including Cohen, Fox, Grosz & Sidner, Hirst, Hobbs, Lockman & Klappholz, Litman & Allen, Mann & Thompson, Polanyi & Scha and Reichman). The summary inspires a composite list of twenty Coherence Relations reproduced in Table 2. Many of these relations (such as Evaluation and Biased Comment) are pragmatic and therefore have no corresponding CLRs in HAIKU.

Sequence	Reported Event
Enablement	Cause
Goal	Parallel
Contrast	Evidence
Elaboration	Generalization
Restatement	Qualification
Evaluation	Description
Situation-Activity	Situation-Time
Situation-Place	Import
Unbiased Comment	Biased Comment

Table 2: Coherence relations from Dahlgren (1988)

Kehler (1993a) hints at a set of Coherence Relationships that includes Contrast, Comparison, Result, etc. In his frame representation for a clause, there is also reference to tense, polarity and modality. The relationships have not been fully enumerated, however, and the potential uses of tense, polarity and modality have not been investigated (Kehler 1993b).

Knott & Dale (1994) describe a method of motivating a set of rhetorical relations (based on the set of relations defined in Mann & Thompson's RST) through inspection of a large corpus. The process involved scanning a corpus for cue phrases (defined as "phrases whose function it is to link spans of discourse together"). These cue phrases (similar to HAIKU's list of CLR Markers) were then used as a basis for classification of relations. The relations are classified according to several binary-valued features including *source of coherence* (semantic or pragmatic), *polarity* (negative or positive), *modal status* (hypothetical or actual) among others (Knott & Mellish 1996). For example, the relation marked by the connective *only when* is semantic, negative, actual, causal.

The system described in Gomez (1995) deals with explanatory connections between two sentences. These relationships can be purely pragmatic (for example in an elaborative relationship) or purely semantic (for example in a causal relationship). The relationships defined in the paper also cover other explanatory connections. For example, the second of two sentences may provide an "explanation of the locations of things" (why do certain things happen to be at certain places?) or a "reaction of animate beings to other animate beings." The kinds of explanatory relationships are given in Table 3.

Completion of thematic roles (Agent, Theme, Recipient, etc.)
Causal relation
Effect or Consequence
Enablement
Reaction of animate beings to other animate beings
Properties
Specification of fuzzy adverbs or adjectives
Explanations of the locations of things

Table 3: Explanatory connections from Gomez (1995)

The system depends on an ontology of world knowledge, and semantic interpretation rules contain further world knowledge, as the following rule (from Gomez 1995) illustrates:

```
if animate(?x) and at-loc(?x,?z) and lack(?z,oxygen) then die(?x)
```

2.1.2 Tense and Modality

Section 2.5 describes how HAIKU assigns CLRs after considering the tense and modality of a clause together with other features. Research on tense and modality was useful in determining how these features relate to the semantics of connected clauses.

Hermerén (1978) presents a list of several semantic modalities and the particular modal auxiliaries that mark them. The semantic modalities are arranged in hierarchies indicating "strength". The modalities higher up in the hierarchy logically imply the modalities below them. For example, Certainty of a proposition implies Prediction of that proposition; Prediction of the proposition implies Probability; Probability implies Possibility, etc.

Palmer (1979) considers modality only as marked by the English modals. He identifies two degrees of modality: Necessity and Possibility, which roughly correspond to the modalities expressed by the stronger and weaker modals in HAIKU.

Coates (1983) lists the semantic modalities expressed by each of the English modals. These modalities express modal "strength" in the labels she gives them (*e.g.* Strong Obligation vs. Weak Obligation; Confident Inference vs. Tentative Inference; etc.). The mapping from the modals to these modalities implies a strong/weak classification of the modals themselves. Quirk *et al.* (1985) give a similar mapping between modal auxiliaries and semantic modalities. That work also makes note of how tense affects modality. For example, the past tenses of modals often express a hypothetical reading of the modalities.

2.2 Input Structures

The input to CLR analysis consists of the *clausal parse trees* and the *logical clausal structure.*

2.2.1 Verb Sequence Features

The clausal parse trees are DIPETT parses corresponding to each clause in a sentence. The parse tree for a finite clause contains detailed syntactic information about the clause, and

particularly about the main verb sequence. Syntactic verb sequence information includes details about the tense, modality and polarity (positive/negative) of the verb sequence. These features will be useful when assigning CLRs to input sentences, as described in detail in section 2.5.

2.2.2 Clausal Organization

There are three variations on the organization of clausal parse trees corresponding to different syntactic configurations of connected clauses. Each format consists of a connective (the lexical item joining clauses) and two or more clause identifiers. In formats where there is more than one coordinator and/or subordinator linking the clauses, the connective used in CLR analysis will be the concatenation of the individual connecting words (for example, *if-then*).

Coordination or Subordination

[*argument1* , *connective* , *argument2*]

The connective linking the two arguments can be either a coordinate conjunction (such as *and*) or a subordinator (such as *until*). Each argument can be either a clause identifier (used as a pointer to the corresponding clausal parse tree) or an embedded clausal structure. Sentence (7) is an example of two clauses—main and subordinate—linked by a subordinator. Following the example is the LCS for the sentence.

(7) *The printer will print until the paper tray is empty.*

```
[ *statement1*, subordinator(until), *statement2* ]
```

Correlative Coordination

[[*coordinator1* , *argument1*] , [*coordinator2* , *argument2*]]

This structure is produced for a sentence with two clauses linked by a correlative such as *either-or*. Although *both-and* and *neither-nor* are also valid correlatives, they cannot be used to correlate whole finite clauses (see Quirk *et al.* 1985: 13.35, 37). Again, each argument can refer to a clause or an embedded logical clausal structure. Sentence (8) is an example of correlative coordination.

(8) *Either the printer is unplugged or the paper tray is empty.*

```
[ [ coord(either), *statement1* ], [ coord(or), *statement2* ] ]
```

Subordinator/Conjunct Correlation

[*coordinator*, [*argument1*, *subordinator*, *argument2*]]

This third format corresponds to clauses linked by a two-part subordinator (such as *if-then*; see Quirk *et al.* 1985: 8.145).

(9) *If the paper tray is empty, then the printer will not print.*

```
[ coord(if), [ *statement1*, subordinator(then), *statement2* ] ]
```

2.3 The Clause Level Relationships

For each pair of clauses in a given logical clausal structure, HAIKU must assign a CLR that captures the semantic relationship between the clauses. Table 4 lists the CLRs with their abbreviations.

CAUSAL	
Causation (CAUS)	Enablement (ENAB)
Detraction (DETR)	Entailment (ENTL)
Prevention (PREV)	
TEMPORAL	
Co-occurrence (CTMP)	Precedence (PREC)
CONJUNCTIVE	
Conjunction (CONJ)	Disjunction (DISJ)

Table 4: The clause level relationships (with abbreviations)

Earlier versions of the set of CLRs were refined by comparing them to lists of semantic relationships previously presented in linguistics and computational linguistics—usually, in discourse analysis. Traditional *discourse relations* deal not only with the semantic relationships between the events represented by clauses, but also with the rhetorical functions of the clauses themselves. The CLRs presented here only deal with semantic relationships, since discourse analysis is not part of HAIKU.

The CLRs are divided into three groups according to the type of relationship they represent: causal, temporal and conjunctive. This division corresponds to a ranking: the causal CLRs imply a temporal ordering (a cause temporally precedes its effect) and also a conjunctive relationship (Prevention implies Disjunction; the other causal CLRs imply Conjunction). Similarly, the temporal CLRs imply Conjunction. In the absence of any other information, this ranking can be used as a default to prefer one CLR over another. For example, when trying to decide whether to assign Causation or Precedence to a given sentence, the system could confidently assign Precedence. Since Causation implies Precedence, Precedence would not be wrong, even if Causation would have been better. I refer to this as *conservative* CLR assignment.

Alternatively, the system could be *aggressive* and assign Causation. Since Causation implies Precedence, Causation is more informative. A comparative evaluation of aggressive and conservative CLR assignment appears in section 2.6.2.

The causal CLRs are all binary. The temporal and conjunctive CLRs can have more than two arguments, but the only connectives that can mark these n-ary CLRs ($n > 2$) are the coordinators *and* and *or*. Of these two, *or* unambiguously marks Disjunction when connecting more than two clauses, while *and* can mark Conjunction, Co-occurrence or Precedence. The binary CLRs present more difficult disambiguation problems.

Most of the CLRs are directed relationships: the order of the clauses involved in the relationship is relevant. This order does not always correspond to the surface order of the clauses in the sentence (see section 2.4). The paraphrases in the following section show the argument ordering for each of the CLRs for which the order is relevant.

Implementation Note

Aggressive/conservative is a parameter that can be set in user profiles for HAIKU. The user can choose from multiple profiles at the start of a session. If CLR assignment is set *aggressive* the system will prefer temporal CLRs over conjunctive and causal CLRs over temporal *in the absence of other evidence for preference.* The user can switch from *aggressive* to *conservative* (or vice versa) at any time during a HAIKU session.

2.3.1 CLR Glossary

In the CLR definitions and paraphrases in this section, E1 and E2 refer to events (as defined in section 1.1.2) or states expressed by clauses. They correspond to the *first* and *second* arguments of directed CLRs. Where the distinction is relevant, arguments are enclosed in square brackets with subscripts identifying E1 and E2. The n-ary CLRs may involve more events, which I have identified as E3... in the paraphrases.

Causal CLRs

Causation (CAUS)

The Causation relationship represents the situation when E1 makes E2 occur or exist. E1 is sufficient to cause E2 and the occurrence or existence of E1 is required.

E1 *causes* E2

(10) [The file printed $_{E2}$] because [the program issued a print command $_{E1}$]. (CAUS)

Detraction (DETR)

The Detraction relationship represents the situation when E1 detracts from or opposes E2 but is insufficient to prevent E2 from occurring or existing.

E1 *detracts from* E2

(11) Although [the server was very busy $_{E1}$], [the program ran $_{E2}$]. (DETR)

Enablement (ENAB)

The Enablement relationship represents the situation when E1 makes E2 possible. E1 is necessary to enable E2 but is not sufficient.

E1 *enables* E2

(12) [The printer can print $_{E2}$] if [the paper tray contains paper $_{E1}$]. (ENAB)

Entailment (ENTL)

For the Entailment relationship, if E1 exists or occurs then E2 must also exist or occur. Unlike Causation, however, E1 is not necessarily known to exist or occur.

E1 *entails* E2

(13) [The printer will print $_{E2}$] if [a print command is issued $_{E1}$]. *(ENTL)*

Prevention (PREV)

A Prevention relationship exists when E1 is meant to keep E2 from occurring or existing. E1 is sufficient to prevent E2.

E1 *prevents* E2

(14) [The files were not copied $_{E2}$] since [the hard disk crashed $_{E1}$]. *(PREV)*

Temporal CLRs

Co-occurrence (CTMP)

Co-occurrence represents the relationship in which events occur or exist at the same time. The time may be a single point on the time line or it may have extent.

E1 *co-occurs with* E2 *co-occurs with* E3...

(15) A job can run in the background while other jobs run in the foreground. *(CTMP)*

Precedence (PREC)

Precedence represents the relationship in which E1 occurs or exists (or begins to occur or exist) before E2.

E1 *temporally precedes* E2

(16) [Printouts were faint $_{E1}$] until [I changed the toner cartridge $_{E2}$]. *(PREC)*

Conjunctive CLRs

Conjunction (CONJ)

A Conjunction relationship exists between events or states about which no more can be said than that they both occur or exist.

E1 *is in conjunction with* E2
is in conjunction with E3...

(17) The computer runs applications and the printer prints documents. *(CONJ)*

Disjunction (DISJ)

A Disjunction relationship exists between events or states about which no more can be said than that one or both occur or exist.

E1 *is in disjunction with* E2
is in disjunction with E3...

(18) The program may terminate or it may hang indefinitely. *(DISJ)*

2.4 The CLR Marker Dictionary

When assigning a CLR to a pair of clauses HAIKU must choose from among the nine relationships in Table 4. Knowing which CLRs are associated with the connective between the clauses allows the system to narrow down the set of candidate relationships. The CLR marker dictionary enumerates the CLRs that are associated with each connective, or *CLR marker*.

The first step in constructing the CLR marker dictionary was to find all possible CLR markers. According to DIPETT, valid connectives are the conjunctions and some adverbial conjuncts (for example *consequently*). The list of markers was taken from DIPETT's dictionary and augmented with conjunctions found in online wordlists and lists of discourse markers (such as the list in Knott & Dale 1994).

Implementation Note

The CLR marker dictionary is compiled with HAIKU. CLRs assigned by the user at run time are stored with the marker and accessible to future CLR interactions. Adding new entries and new mappings to existing entries in the CLR marker dictionary is a simple matter of adding Prolog facts to the dictionary in the form shown in Appendix I. New mappings resulting from a HAIKU session can simply be copied straight from the session script to the CLR marker dictionary.

The next step was to map each marker sense to the appropriate CLRs. These marker→CLR mappings were determined by studying the conventional dictionary[1] senses of each connective to learn what semantic relationships they represent. For each sense, an entry was added to the dictionary mapping the marker to the appropriate CLR. Appendix I gives a sample of the entries in the CLR marker dictionary.

Although the marker dictionary has been constructed carefully, it is possible that it is incomplete. If there is a CLR missing for a known connective, the CLR analyzer will not consider that CLR as a candidate and the user will have to supply it. If the system encounters an unknown connective, all of the CLRs are candidates, and again the user will have to supply the correct one. The system does have mechanisms to recover from these deficiencies so that the user does not have to keep entering the same information (see section 2.5.5).

Mapping the senses to CLRs uncovered a consistent *direction* for each connective. As stated in 2.3, for most of the CLRs the order of the arguments is relevant. For example, for Entailment, one clause is the *antecedent* and the other is the *consequent*. With some connectives, the clause introduced by the connective is the antecedent, as in (19). With other connectives the clause introduced by the connective is the consequent, as in (20). This correspondence between the syntactic positions of the clauses and the ordering of semantic arguments of the CLR is consistent for each connective/CLR pair. The direction is stored with each marker→CLR mapping in the CLR marker dictionary so that CLR analysis can determine automatically the correct ordering of semantic arguments, given the order of the syntactic arguments.

(19) *[The file will print* $_{consequent}$*] if [the program works* $_{antecedent}$*].*

(20) *[The program works* $_{antecedent}$*] so [the file printed* $_{consequent}$*].*

---

[1] I used *COBUILD* (Sinclair 1991), the unabridged *Random House* (Stein 1983), *LDOCE* (Summers 1987) and Quirk *et al.* (1985) to find marker senses.

2.5 Assigning CLRs

CLR analysis assigns a relationship to a pair of arguments. Each argument is either a clausal parse tree or an embedded CLR structure. For embedded structures, analysis begins with the innermost nested pairs (or sequences for n-ary CLRs) of clauses and proceeds to the outermost relationships. Consider sentence (21) and its corresponding logical clausal structure.

(21) *The printer can print if the program issues the print command before the system shuts down.*

```
[*statement1*, subordinator(if),
      [*statement2*, subordinator(before), *statement3*] ]
```

The clause identifier `*statement1*` refers to the subtree for *the printer can print*; `*statement2*` to the subtree for *the program issues the print command* and `*statement3*` to *the system shuts down*. CLRA first assigns a CLR to the relationship between `*statement2*` and `*statement3*` resulting in CLR structure *S*. CLRA then assigns a CLR to the relationship between `*statement1*` and *S*.

The first step in assigning a relationship to a pair of arguments is to consult the CLR marker dictionary for the CLRs marked by the current connective. From these candidates, CLRA chooses one or more best CLRs for the relationship after considering the syntactic features described below. The system then submits the best CLRs to the user for approval. The user can accept one of the suggested CLRs or reject the suggestions and enter any other CLR. Since the user may enter any CLR, including one not known to be marked by the current connective, HAIKU should keep a record of user assignments to avoid making the same mistakes and forcing the user to enter information already entered. Section 2.5.5 describes the storage and recall of user input to assist CLR assignment.

2.5.1 Verb Phrase Polarity and Connective Polarity

The polarity (positive or negative) of the verb phrase in a clause can be used to determine whether a positive causal CLR (Causation, Enablement, Entailment) or a negative causal CLR (Detraction, Prevention) is more appropriate for a given input sentence. For

example, often the second semantic argument (E2) has a positive verb phrase for Entailment but a negative verb phrase for Prevention:

(22) [The document will not print $_{E2}$] if the paper tray is empty. *(PREV)*

(23) [The document will print $_{E2}$] if the printer is ready. *(ENTL)*

With certain connectives verb phrase polarity has the opposite effect in determining the CLR. That is, for some connectives a *positive* E2 implies Prevention and a *negative* E2 implies a positive causal CLR:

(24) [The document will print $_{E2}$] unless the paper tray is empty. *(PREV)*

(25) [The document will not print $_{E2}$] unless the printer is ready. *(ENAB)*

I use the term *connective polarity* to refer to the feature of a connective that affects how verb phrase polarity determines the CLR. Connectives such as *unless* that reverse the effect of verb phrase polarity in determining the CLR will be called *negative* connectives. This terminology is not meant to suggest that the connective itself is inherently positive or negative. Rather, it reflects the connective's relationship to verb phrase polarity in determining whether the CLR is positive or negative. Other negative connectives include *although*, *but*, *either-or*, *except*, *only*, *or*, *save that*, *until*, *yet*.

The examples also illustrate the need for care in producing CLRA output. For sentence (22), CLRA output should be

the paper tray is empty `prevents` *the document will print*

with the negation operator (*not*) deleted. The negation should also be removed from sentence (25):

the printer is ready `enables` *the document will print*

The following *polarity reversal rule* formalizes the required action for producing CLRA output:

For causal CLRs reverse the polarity of the second CLR argument in CLRA output whenever there is a:
a) negative CLR marked by a positive connective, or
b) positive CLR marked by a negative connective

Note that polarity is often implicit in one of a clause's elements. Compare clauses (26), (27) and (28):

(26) the program will not succeed

(27) the program will fail

(28) the program will experience failure

In clause (26), the polarity is explicit and can be used to assist CLR analysis. Clause (27) has an implicitly negative verb while clause (28) has an implicitly negative complement. Identifying these clauses as negative for CLR analysis would require the semantic knowledge that *fail* and *failure* are implicitly negative. HAIKU does not have access to this kind of information. The absence of such lexical knowledge, however, will not result in an incorrect interpretation:

(29) The program will fail to terminate if there are bugs.

Sentence (29) will be interpreted as

there are bugs `entails` *the program will fail to terminate*

This interpretation is valid, if not as elegant as

there are bugs `prevents` *the program will terminate*

2.5.2 Verb Phrase Tense and Modality

In a purely surface-syntactic analysis of a clause, it is often impossible to distinguish modals from auxiliaries inflected for tense. For example, in sentence (30), the word *could* is used as the past tense of the modal of ability *can*, whereas in (31), *could* is used as a conditional auxiliary.

(30) Victor could stand on his head when he was seven.

(31) Roy could do that if he wanted to.

Similar tense/modality ambiguities exist with *may*, *will* and others.[2] These ambiguities suggest that CLRA should consider tense and modality together.

The tenses and modalities of the clauses in a sentence often hold clues to the semantic relationships between clauses. The difference between many of the causal CLRs is the degree of certainty of the consequent (E2). For example, the difference between Entailment and Enablement is that Entailment implies a high degree of certainty that the consequent will occur; Enablement, a much weaker certainty.

Research on the English modals has identified the concept of *modal strength*. Palmer (1979), Coates (1983) and Quirk *et al.* (1985) divide the modals into two broad categories: those denoting some degree of *necessity* (stronger modals) and those denoting some degree of *possibility* (weaker modals). Hermerén (1978) goes further to suggest that the modals represent distinct degrees of modality. He offers the ranking *certainty→prediction→probability→possibility*. The definition of HAIKU's causal relationships as sufficient or insufficient suggests that a binary division into stronger modals and weaker modals is appropriate for disambiguating the CLRs.

Table 5 shows the modals recognized by DIPETT, divided into stronger and weaker modals. Some notes on the division are in order.

STRONGER MODALS	WEAKER MODALS
cannot	can
dare not	could
must	may
need	might
shall	need not
will	ought
would	should

Table 5: Strength of the modals

[2] Lyons (1995) discusses the overlap between futurity and modality, noting that in English and other languages, many uses of future tense are modal rather than temporal.

Absence of Modals

Since modals represent some degree of uncertainty, a clause with no modal auxiliary will always be considered stronger than a clause with a modal.

Modals in Negative Verb Phrases

Some modals in negative verb phrases signal an opposite semantic modality to that expressed by the same modal in positive verb phrases (see Hermerén 1978). For example, the modal *can* represents possibility when appearing in positive verb phrases (32) but represents necessity in negative verb phrases (33):

(32) The program can print.

(33) The program cannot print.

On the stronger/weaker scale, *can* would be a weaker modal while *cannot* would be a stronger modal.

The modal *may* is unusual with respect to negative verb phrases. *May* only becomes a stronger modal when appearing in a negative verb phrase with a second person subject where it sometimes becomes a modal of permission.

(34) You may not leave.

Sentence (34) is unambiguous between a stronger sense of permission and a weaker sense of intention: language users do not tend to make assertions about an addressee's intent.[3] Therefore, *may not* with a second person subject is a stronger modal. In technical texts, however, the use of the second person as a generic pronoun is common.

(35) If network traffic is heavy, you may not get a line.

---

[3] It is possible to imagine a conversational usage of *may* with a second person subject marking a weaker intention modality: Michael: *I won't be able to help if I go to Moose Jaw*; Michele: *Yes, but you might be able to help because you may not be leaving until Friday*. Such a usage is unlikely to appear in declarative technical texts.

Example (35) shows the second person with a negated *may* as a weaker modal of possibility. Because the use of *may not* as a stronger modal of permission is inconsistent—especially in technical text—it does not appear with the stronger modals in Table 5. The strength of the remaining modals does not change when they are negated.

Marginal Modals

The marginal modals in Table 5 are *dare*, *need* and *ought*. Marginal modals are considered similar to central modals in function, but they usually appear in patterns where they more closely resemble main verbs than auxiliaries (see Quirk *et al.* 1985: 3.40-43).

Dare as a modal only occurs in modern English in negation (see Palmer 1979: 27). The negated *dare* in sentence (36) is a stronger modal, as supported by its paraphrase in (37).

(36) I dare not go.

(37) It is certain that I will not go.

Any remaining forms of *dare* as a modal in the positive, such as *I dare say* have become idiomatic and would occur rarely (if ever) in technical texts.

Similar to *dare*, *need* as a modal usually only occurs in modern English in negation. The negated *need* in sentence (38) is a weaker modal as supported by the paraphrase in (39). In the rare situation where *need* occurs in the positive—as in (40)—it is a stronger modal.

(38) I need not go.

(39) It is uncertain that I will (not) go.

(40) Only software engineers need apply.

Should *as a Past Tense of* **Shall**

The modal *should* can be interpreted as the past tense of *shall*, suggesting that it is a stronger modal (as in example (41) taken from Quirk *et al.* 1985: 4.58).

(41) I felt sure that we should meet again.

Should as a past tense of *shall* is rare and its use is restricted to indirect speech. The more common interpretation of *should* is the tentative, weaker modal use in example (42).

(42) The program should make backups automatically.

2.5.3 Using Argument Features to Choose CLRs

When trying to find the most appropriate CLR among candidates, the system will perform many pairwise comparisons of CLRs. For each pair of CLRs, the polarity, tense and modality features of the clauses will determine if one CLR is more appropriate for the given input than another.

In this section I look at all possible pairs of CLRs (36 such pairs for 9 CLRs) to determine how they relate to polarity, tense and modality. The result is a set of preference rules for distinguishing CLRs, based on the syntactic features of the clauses. Using these rules, the system will suggest a CLR assignment to the user, who can accept or reject the suggestion.

For CLRs that seemed potentially ambiguous but had no common connective, I revisited the CLR marker dictionary to see if there were missing entries. Several such holes in the marker list were discovered and entries added.

For some pairs of CLRs, the syntactic features of the arguments are not consistently different enough to allow disambiguation. These ambiguous pairs have an effect on the outcome of CLR competitions, described in section 2.5.4. Moreover, for many of the pairs there are no connectives known to mark both. If the user ever assigns one CLR from such a pair to a connective that marks the other CLR, the rules will not be able to disambiguate between them in future processing. This drawback underlines the importance of recording user assignments to assist future processing (see section 2.5.5). Table 6 lists the pairs of CLRs that share no markers. These pairs are omitted from the pair-by-pair discussions that follow.

Causation and Conjunction	Entailment and Disjunction
Causation and Disjunction	Prevention and Conjunction
Detraction and Conjunction	Prevention and Precedence
Detraction and Disjunction	Co-occurrence and Precedence
Detraction and Precedence	Co-occurrence and Disjunction
Enablement and Conjunction	Precedence and Disjunction
Enablement and Disjunction	Conjunction and Disjunction
Entailment and Conjunction	

Table 6: Pairs of CLRs that share no markers in the CLR marker dictionary

Of the pairs of CLRs that do share markers, some pose difficult disambiguation problems. For example, it is difficult in general to distinguish the causal and temporal CLRs when marked by the same connective:

(43) Don's car has stalled since it was low on gas. *(CAUS)*

(44) Bob has slept since the game started. *(PREC)*

The presence of modals might signal a preference for the causal CLRs, but some causal CLRs (like Causation) are distinguished by an absence of modals. Moreover, the ambiguity between modal auxiliaries and tense indicators makes a distinction even more difficult.

Lacking a compelling theoretical argument for distinguishing these types of CLRs, it may be useful to consider a practical argument. Any module that makes use of HAIKU output could probably infer more from causal CLRs than from temporal or conjunctive CLRs.[4] This performance preference for choosing causal CLRs corresponds to the aggressive CLR assignment defined in section 2.3. Of course, since the CLR analyzer is semi-automatic, the user will always have the option of rejecting inappropriate suggestions. Table 7 shows pairs of CLRs that cannot be disambiguated using verb phrase features. If CLR assignment is set aggressive, HAIKU will prefer the causal CLRs over the temporal ones.

[4] For example, a module attempting to learn from CLR structures could construct a *rule* from causal CLRs. Given a conjunctive CLR, it may only be able to assert the CLR arguments as unrelated facts.

Causation and Co-occurrence
Causation and Precedence
Enablement and Co-occurrence
Enablement and Precedence
Entailment and Co-occurrence
Entailment and Precedence
Prevention and Co-occurrence

Table 7: Pairs of CLRs that cannot be disambiguated using verb phrase features

CLR Preference Rules

This section contains rules for using verb phrase features to choose between the remaining pairs of CLRs. As usual, where there is a distinction between the first and second CLR arguments (corresponding to E1 and E2 in section 2.3.1), the arguments are bracketed and identified with subscripts.

Causation vs. Detraction

For Causation, the occurrence of the consequent is required. Logically, Causation is an implication where the antecedent is known to have occurred. For this reason, modals are unlikely to appear in either E1 or E2 (since modals, by definition, express a degree of uncertainty; see Quirk *et al.* 1985: 4.49).

For Detraction, the occurrence of the consequent is uncertain, and E2 is more likely to contain modals—usually weaker modals. The polarity of E2 may also be used as a distinguishing feature (as with Causation vs. Prevention). Although a negative consequent is not a required feature for Detraction, it is unlikely to appear with Causation.

(45) *[The program worked $_{E2}$] because there was sufficient memory.* *(CAUS)*

(46) *[The program may not work $_{E2}$] because memory is low.* *(DETR)*

Implementation Note

Since it is possible to add marker→CLR mappings to the CLR marker dictionary (and conceivable to add CLRs), the set of preference rules may be incomplete. The implementation of the CLR competition module (section 2.5.4) does not require that the preference rules be complete. In the absence of a preference rule for a given pair of CLRs, the system gives each equal weight in determining the most appropriate CLR. Other factors, such as previous user input and preference rules with *other* CLRs will also affect the ultimate choice of CLR.

Causation vs. Enablement

For Enablement, the occurrence of E2 is uncertain, since the enabling clause is insufficient to guarantee E2. Weaker modals are expected in E2 for Enablement while no modals are expected for Causation.

(47) [The program stopped $_{E2}$] because the machine was out of memory. (CAUS)

(48) [The program should run $_{E2}$] because the machine has enough memory. (ENAB)

Causation vs. Entailment

Again, the occurrence of the consequent is required for Causation. For Entailment, the consequent is contingent on the truth of the proposition represented by E1. Modals (in particular stronger modals) may be common in E2 for Entailment, but should not appear if the relationship is Causation.

(49) The program failed, therefore [the system crashed $_{E2}$]. (CAUS)

(50) The program failed, therefore [the system will crash $_{E2}$]. (ENTL)

Causation vs. Prevention

When Causation and Prevention are marked by the same connective, E2 has a positive polarity for Causation whereas for Prevention the polarity of E2 is negative. Note that only condition a) of the polarity reversal rule will apply since Causation is never marked by a negative connective.

(51) [We adjourned the meeting $_{E2}$] as it was getting late. (CAUS)

(52) [We could not continue the meeting $_{E2}$] as it was getting late. (PREV)

Detraction vs. Enablement

By definition, Enablement is an insufficient positive causal CLR and Detraction is an insufficient negative causal CLR. Both CLRs are likely to have weaker modals in E2. For a given sentence, if E1 and E2 have different polarities with a positive connective or the same polarities with a negative connective, Detraction should be chosen. Otherwise, verb phrase features do not distinguish Detraction and Enablement.

(53) If the display does not work, [the printer can still print files $_{E2}$]. (DETR)

(54) If the power is on, [the printer can quickly print files $_{E2}$]. (ENAB)

Detraction vs. Entailment

Entailment tends to have stronger modals in E2, Detraction tends to have weaker modals.

(55) If the system is unresponsive, [the computer may still be working $_{E2}$]. (DETR)

(56) If the system responds, [the computer must be functioning $_{E2}$]. (ENTL)

Detraction vs. Prevention

The consequent of Prevention must be negative (perhaps implicitly) when marked by a positive connective; the consequent of Detraction need not be negative. If the consequent is negative with a positive connective (or positive with a negative connective) or if the required negation is implicit, the verb phrase features do not distinguish these two CLRs.

(57) If the system is unresponsive, [the computer will still work $_{E2}$]. (DETR)

(58) If the system is hung, [the computer will not work $_{E2}$]. (PREV)

Detraction vs. Co-occurrence

The only marker marking both Detraction and Co-occurrence is *while*, which seems more commonly temporal. Co-occurrence should be preferred (overriding the aggressive preference for causal CLRs over temporal CLRs).

(59) The program suggests one interpretation while the user prefers another. (DETR)

(60) The backup continues while normal processing takes place. (CTMP)

Enablement vs. Entailment

For Enablement E2 tends to have weaker modals whereas for Entailment E2 is likely to contain stronger modals.

(61) If the power is on, [the computer can work $_{E2}$]. (ENAB)

(62) If the program responds, [the computer must be working $_{E2}$]. (ENTL)

Enablement vs. Prevention

These two CLRs differ both in strength of modal in E2 (Enablement—weaker modal; Prevention—stronger modal) and in the polarity of E2 and the connective.

(63) If the power is on, [the printer can be used $_{E2}$]. *(ENAB)*

(64) If the power is off, [the printer will not print $_{E2}$]. *(PREV)*

Entailment vs. Prevention

Although Entailment and Prevention are both sufficient causal CLRs, they differ in polarity. With a positive connective Entailment should have a positive consequent and Prevention should have a negative consequent. If the connective is negative, Entailment should have a negative consequent and Prevention should have a positive consequent. The polarity reversal rule applies in all cases.

(65) If the printer has paper, [the file will print $_{E2}$]. *(+E2, +conn: ENTL)*

(66) If the printer is out of paper, [the file will not print $_{E2}$]. *(-E2, +conn: PREV)*

(67) Unless the printer has paper, [the file will not print $_{E2}$]. *(-E2, -conn: ENTL)*

(68) Unless the printer is out of paper, [the file will print $_{E2}$]. *(+E2, -conn: PREV)*

After application of the polarity reversal rule, the desired CLR interpretation for (65) and (67) will be:

the printer has paper `entails` *the file will print*

The interpretation for (66) and (68) will be:

the printer is out of paper `prevents` *the file will print*

Prevention vs. Disjunction

The only connectives that mark both Prevention and Disjunction are *or* and *either-or*. Since there are no sufficient syntactic distinctions between their respective arguments, a pragmatic choice may be made. Since Disjunction is only marked by *or* and *either-or* (whereas Prevention is marked by several other markers), Disjunction will be preferred over Prevention.

(69) The program must terminate or the system will crash. *(PREV)*

(70) The program will terminate or it will fail to terminate. *(DISJ)*

Co-occurrence vs. Conjunction

There is little syntactic information to distinguish Co-occurrence and Conjunction. Since Conjunction is relatively bland, Co-occurrence will be the preferred suggestion.

(71) The program ran and the user waited. *(CTMP)*

(72) The program ran and the printer printed. *(CONJ)*

Precedence vs. Conjunction

The only marker Precedence and Conjunction have in common is *and*, which is less commonly used for Precedence. Conjunction should be preferred.

(73) The program computed the value and it terminated. *(PREC)*

(74) The program ran and the printer printed. *(CONJ)*

2.5.4 CLR Competitions

To choose a single CLR from a set of candidates HAIKU uses the preference rules from the previous section. Each of those rules is implemented as a heuristic for choosing between a given pair of CLRs. If there are more than two candidates, the rules must be applied to individual pairs within the set of candidates. Since the rules do not always distinguish CLRs, the system must also allow for ties.

The CLR analyzer gathers the candidate CLRs from the CLR marker dictionary. Each candidate CLR competes against all other candidates. Each time the preference rules prefer one candidate over another, the winner collects two points, while the loser collects no points. If the rules do not prefer one candidate over another, the competition is declared a tie and each candidate receives a single point. Once all competitions have been held, the CLR with the most points is presented to the user for approval as the most appropriate CLR for the given input.

Example (75) will illustrate the CLR competition model:

(75) *Since [micrometers can measure so accurately $_{E1}$]*
[they can be used to detect the slightest wear on engine parts $_{E2}$]

The positive connective *since* is listed in the CLR marker dictionary mapped to five CLRs: Causation, Enablement, Entailment, Prevention and Precedence. HAIKU holds ten competitions between these candidate CLRs. The outcomes of each competition are shown in Table 8 along with brief explanations of why the winner won. The final victor suggested to the user is Enablement, which scored eight points (the maximum for five competitors). The points accumulated by each CLR are shown in Table 9.

CLR1	*CLR2*	*Winner*	*Reason*
Causation	Enablement	Enablement	the modal *can* appears in E2
Causation	Entailment	Entailment	the modal *can* appears in E2
Causation	Prevention	Causation	the polarity of E2 is positive
Causation	Precedence	Causation	aggressive bias toward causal CLRs
Enablement	Entailment	Enablement	the weaker modal *can* appears in E2
Enablement	Prevention	Enablement	E2 is positive and has a weaker modal
Enablement	Precedence	Enablement	aggressive bias toward causal CLRs
Entailment	Prevention	Entailment	positive connective with positive E2
Entailment	Precedence	Entailment	aggressive bias toward causal CLRs
Prevention	Precedence	Prevention	aggressive bias toward causal CLRs

Table 8: Outcomes of individual CLR competitions for (75)

2.5.5 Using User Assignments to Choose CLRs

The CLR analyzer performs reasonably well (see section 2.6) using only the syntactic features described above to guide assignment. When the system makes incorrect suggestions, the user is expected to supply the correct assignment. HAIKU records the user's assignment to avoid making the same incorrect suggestions the next time it sees the same (or a similar) input.

	wins	*losses*	*ties*	*points*
Causation	2	2	0	4
Enablement	**4**	**0**	**0**	**8**
Entailment	3	1	0	6
Precedence	0	4	0	0
Prevention	1	3	0	2

Table 9: Points accumulated by each CLR in competitions for (75)

CLR Assignment Attributes

Associated with every CLR assignment are a connective, two clauses and a CLR. Each clause has two syntactic features: tense/modality and polarity. The connective, the syntactic clausal features and the CLR together make up a pattern of attributes of a CLR assignment:

Connective	*T1*	*P1*	*T2*	*P2*	*CLR*

Connective is the CLR marker. *T1* and *T2* are the tense/modality values for the first and second clausal CLR arguments, E1 and E2. *P1* and *P2* are the polarity values for E1 and E2. *CLR* is one of the nine CLRs. The CLR analysis problem is to assign a value to *CLR*, given the values of the other five elements of the pattern. I will refer to the pattern as an *attribute pattern.*

After the user has approved a CLR assignment, HAIKU stores an instance of the attribute pattern with the assigned CLR. The system can then use matching attributes from stored patterns as an alternative to using the preference rules and competitions.

For example, using the competition model, the system would suggest Enablement for (76) because of the weak modal *may* and the bias for causal CLRs over temporal.

(76) *[The exhaust port may still be open* $_{E2}$*] when [the piston uncovers the intake port* $_{E1}$*].*

For this sentence, however, the user should supply Co-occurrence as more appropriate. The resulting attribute pattern would be stored with the Co-occurrence CLR:

when	present_simple/ no_modal	pos	may_present/ weak_modal	pos	ctmp

If the system then encounters another sentence with the same attribute values, it can suggest Co-occurrence based on the stored pattern for (76).

When to Use User Assignments

For a given input, any or all of the attributes may be relevant to the choice of a CLR. If the system requires perfect matches with previous attributes, it assumes that all attributes were relevant in the previous assignment and are relevant to the current assignment.[5] In this section I describe an algorithm that allows partial matching of previous CLR assignments to guide CLR analysis. Section 2.6.3 evaluates the implementation of these techniques in HAIKU's CLRA module.

The algorithm has two parts: one part that stores attribute patterns for future analyses, and one part that matches attribute values of the current input to those in stored patterns.

Storing Attribute Patterns

When the user accepts one of the system's suggestions for a given input, that input's attribute pattern is stored with the accepted CLR. If the user rejected the system's suggestions and supplied the CLR, the pattern is stored as a user assignment of the supplied CLR. When storing user assignments of a CLR, the system checks all previous stored patterns having the same connective and CLR. HAIKU then stores a *least-general generalization* (defined below) of the current pattern with each of the most closely matching stored patterns.

Assigning CLRs Based on Stored Attribute Patterns

For a new input, the system gathers all stored attribute patterns that have all five attributes matching the input.[6] It then builds a list of the CLRs associated with the stored patterns and chooses the most frequent CLRs as most likely appropriate for the sentence. If there are no perfectly matching stored patterns, HAIKU proceeds with the preference-based competitions from section 2.5.4.

---

[5] On the other hand, if the system allows partial matches, it may overgeneralize and ignore relevant attributes, resulting in incorrect suggestions. A large experiment could provide statistical evidence of correlation between features and CLR assignments, thereby identifying relevant features. The sparseness of CLR data (see section 2.6) suggests that thousands more parses of complex sentences are needed before such an experiment could be done.

[6] Generalized attributes perfectly match any value.

Checking stored attribute patterns before running competitions is a bias in favour of previous analyses over the preference rules. Since patterns are also stored for assignments resulting from successful CLR competitions, however, the preference rule knowledge is not lost. The preference rules resulting in correct CLR assignments for the text persist in the form of stored attribute patterns (see section 5.1.4).

Least-General Generalization of Attribute Patterns

Let **V** be the attribute pattern to be stored for the most recent CLR assignment. HAIKU stores **V** as-is with its assigned CLR. If **V**'s CLR assignment was user-supplied, the system finds a set of stored attribute patterns $\mathbf{S_{max}}$ with the same connective and same assigned CLR. Each pattern in $\mathbf{S_{max}}$ has a maximum number of matching attributes—that is, for each pattern $\mathbf{V'}$ in $\mathbf{S_{max}}$, there is no stored pattern outside of $\mathbf{S_{max}}$ that has more matching attributes with **V** than $\mathbf{V'}$.

The *least-general generalization* (*lgg*) of **V** and $\mathbf{V'}$ is an attribute pattern $\mathbf{V_{lgg}}$ defined as follows: for each attribute **A** in **V**, if the corresponding attribute $\mathbf{A'}$ in $\mathbf{V'}$ is equal to **A**, then attribute $\mathbf{A_{lgg}}$ in $\mathbf{V_{lgg}}$ is equal to **A**. If **A** and $\mathbf{A'}$ are different, then $\mathbf{A_{lgg}}$ is a variable that will match any value of **A** in future patterns.

HAIKU computes the lgg of **V** with each $\mathbf{V'}$ in $\mathbf{S_{max}}$ individually and stores the resulting generalized patterns for future analyses.

For example, assume the system has already stored the following attribute patterns:

1	when	present_simple/ no_modal	pos	present_simple/ no_modal	pos	ctmp
2	when	present_simple/ no_modal	pos	future_simple/ strong_modal	pos	entl

Suppose now that the user has just made a CLR assignment resulting in the attribute pattern:

3	when	present_simple/ no_modal	pos	may_present/ weak_modal	pos	ctmp

This pattern will be stored as-is for future processing. HAIKU will then find stored patterns with the maximum number of matching attributes and the same CLR. Both attribute

patterns *1* and *2* match on the connective and three of the other four attributes. Only pattern *1* however has the same assigned CLR. HAIKU computes the lgg of patterns *1* and *3*, and adds the resulting pattern *4* to the set of stored patterns, containing *X* as a variable for the second tense/modality attribute.

4	when	present_simple/ no_modal	pos	*X*	pos	ctmp

Subsequent CLR interactions may match pattern *4* perfectly, even if they have a different value for T2.

Avoiding Overgeneralization

Taking the lgg of the current attribute pattern with all stored patterns would be too permissive in matching patterns during analysis. To keep the proliferation of generalized patterns in check, I have placed the following restrictions on generalization.

- Generalize only attribute patterns with the maximum number of matching fields. If **V** matches some of the stored patterns on three features ($\mathbf{S_3}$) and some on only two ($\mathbf{S_2}$), store only the lgg of **V** and patterns in $\mathbf{S_3}$.

- Do not generalize on attribute patterns that match on Connective and CLR and no other features. The resulting generalizations would be maximally general.

2.6 Evaluation

This section summarizes several CLR analyzer experiments on three texts: *building code*, *clouds* and *small engines*. The results of all experiments are presented as the number of each of the three kinds of user action described in section 1.4.3. The first user action is *accept*: HAIKU presented the user with one single CLR and that CLR was correct. The second kind of action is *choose*: HAIKU presented multiple CLRs and the correct CLR was among them. The third action is *supply*: the correct CLR was not among HAIKU's suggestions so the user had to supply the correct relationship.

2.6.1 Diagnostic Evaluation

The first experiment applied CLRA to 100 sentences from the *building code*. In order to exercise the preference rules and the competition model, I chose sentences containing clauses connected by a variety of connectives known to mark two or more CLRs (from the CLR marker dictionary). Apart from this restriction, the sentences were chosen randomly from the entire text. Forcing the system to handle a variety of ambiguous connectives provided good exercise for the preference rules, but the distribution of syntactic phenomena in this data set is not representative of their distribution in complete texts. This experiment, therefore, is more of a diagnostic evaluation (see section 1.4) of the system than the performance evaluations described in section 2.6.2 and 2.6.3.

For each of the 100 sentences, DIPETT provided the tense, modality and polarity features of each clause. The CLR analyzer then held competitions to determine the most appropriate CLR based on these features and the preference rules. CLR attribute patterns were not stored. The percentages of each kind of user action appear in Table 10 along with the number of sentences for which the user had to reverse the CLR arguments because HAIKU ordered them incorrectly.

accept	*choose*	*supply*	*reorder*
94%	4%	2%	2

Table 10: CLRA user actions required in the building code *experiment*

The high success rate for the *building code* experiment suggests that the syntactic features contain enough information for the system to choose a single CLR for most of the sentences. Furthermore, for the majority of sentences, the rules chose the correct CLR. No new CLRs were needed to account for the test sentences.

2.6.2 Performance Evaluation

The experiments described in this section applied CLRA to all of the parse trees in the *clouds* text and *small engines* text. Sentences were not chosen specifically to reflect a variety of syntactic phenomena.

System Performance

The *clouds* experiment tested the CLR analyzer on the 512 sentences in that text. 51 of the sentences contained multiple clauses suitable for CLR analysis. Also measured in this experiment were user onus ratings as described in section 1.4.4. For 50 of the 51 sentences, CLRA interaction was simple (user onus 0), requiring only a few seconds of thought to arrive at the correct decision. Only one sentence required a few extra moments of reflection and consultation of the CLR definitions (user onus 1). Again, no new CLRs were needed to account for the relationships in the test text.

The proportion of correct analyses in the *clouds* experiment was 69%, considerably lower than the corresponding number in the *building code* experiment. The experiment revealed certain shortcomings of the CLR analyzer. First, CLRA's treatment of sentence-initial coordinators resulted in a misinterpretation of sentences (77) and (78) as containing two-part correlative connectives (see section 2.2.2) *and-if* and *but-when*. The leading conjunctions are inter-sentential connectives that should not be used for CLR analysis.

(77) And if everything is just right, you may see a rainbow.

(78) But it's no fun for the farmer when hail hits his crops.

The poorer performance of the CLR analyzer in the *clouds* experiment may also be due to the relative infrequency of complex verb features in the text, which uses simple tenses and few or no modals. To compound the problem, where modality is expressed, it is often expressed using non-auxiliary modal forms, such as the adjective *apt* in sentence (79) and the adverb *usually* in (80).

(79) As clouds change, the weather is apt to change.

(80) When lightning strikes, it usually hits the high pointed objects.

Finally, CLRA did not attempt to determine the correct order of the CLR arguments for user-supplied CLRs. The ordering heuristic, however, depends only on the connective and the CLR. Once the user has entered the appropriate CLR, the system has enough information to determine the correct argument order by checking the mapping from the given connective to the supplied CLR in the CLR marker dictionary. If there is no such

mapping (*e.g.*, if the user enters a CLR not in the marker dictionary for the given connective), the system can guess at a direction by looking at the connective→CLR mappings for CLRs that *do* appear in the marker dictionary for the given connective.

The results from the *clouds* experiment inspired the following modifications:

- Sentence-initial coordinators (discourse markers) are ignored by CLRA.
- Automatic argument ordering is attempted even for user-supplied CLRs.

The modified system was then tested twice on the same 51 sentences from *clouds*: once with aggressive CLR assignment (as in the original experiment) and once with conservative assignment. The retests showed a slight improvement when keeping aggressive assignment. With conservative CLR assignment the number of correct analyses dropped significantly. Finally, all instances of user-supplied CLRs were correctly ordered automatically. Results of the *clouds* experiment (with the two retests) appear in Table 11.

	accept	*choose*	*supply*	*reorder*
original (aggressive) experiment	35 (69%)	2 (4%)	14 (27%)	7
aggressive retest	37 (73%)	2 (4%)	12 (23%)	0
conservative retest	23 (45%)	2 (4%)	26 (51%)	0

Table 11: CLRA user actions required in the clouds *experiment*

The more complex syntax in the *small engines* text meant that only 21 CLRs were assigned; parse tree errors prevented another 55 pairs of clauses from receiving CLR analysis. Of the 21 interactions in the experiment, 17 were simple (onus 0) while 4 required some reflection (onus 1). Once more, the existing nine CLRs were sufficient to account for the semantic relationships between clauses in the text.

In order to get a better evaluation of CLRA, I retested the *small engines* text supplying CLRA with correct parse information for all 76 sentences containing multiple clauses. Interestingly, performance was similar to the original experiment. Results from the *small engines* experiment and the complete retest appear in Table 12. No manual argument reordering was necessary.

	accept	*choose*	*supply*
original experiment (21 CLRs)	14 (67%)	2 (10%)	5 (24%)
complete retest (76 CLRs)	47 (62%)	10 (13%)	19 (25%)

Table 12: CLRA user actions required in the small engines *experiment*

Coverage

CLRA is the HAIKU module most affected by parse errors. In order to make an assignment, CLRA needs a complete and correct parse at the top-most level. A second problem with CLR analysis is the sparseness of CLR data: every sentence usually has several noun phrases and at least one finite clause, but relatively few have multiple connected finite clauses.

In the *clouds* experiment, there were 76 connected clauses requiring CLR assignments, 67% of which received analysis. For the *small engines* text, only 28% of the 76 clause pairs received analysis.[7] The small number of CLRs actually captured by HAIKU for the *small engines* text is a direct result of the error rate in parsing structurally complex sentences.

2.6.3 Can CLRA Learn?

A result not reflected in the tables in the previous sections is the fact that CLRA tends to repeat its mistakes and cannot recover from missing marker→CLR mappings or unknown connectives. To remedy these problems HAIKU's CLR analyzer has been extended to include the techniques for partial matching on stored assignments to guide CLR analysis, as described in section 2.5.5.

In this section I give the results of retesting the extended CLR analyzer on *clouds* and *small engines*. A comparison of the original tests and the retests appears in Table 13.[8]

---

[7] It is purely coincidental that both the *clouds* text and *small engines* text contained 76 sentences with multiple clauses suitable for CLR analysis.

[8] The numbers for the *clouds* experiment from Table 13 do not quite match those from the second row of Table 11. The extra *accept* action is the result of an addition to the CLR Marker dictionary since the original *clouds* experiment from Barker & Delisle (1996).

		accept	*choose*	*supply*
clouds	*CLR preference rules only*	38 (75%)	2 (4%)	11 (22%)
	CLRA + stored assignments	36 (71%)	2 (4%)	13 (25%)
small engines	*CLR preference rules only*	47 (62%)	10 (13%)	19 (25%)
	CLRA + stored assignments	51 (67%)	9 (12%)	16 (21%)

Table 13: CLRA user actions required with partial matching on stored assignments

For the *clouds* experiment, the number of CLRs correctly determined by the system actually decreased when allowing suggestions based on stored assignments. As already observed, most of the clauses in the *clouds* text are present tense and modals are rare. In the *small engines* text there is much greater variety in tense and modality, which may account for the slight improvement in performance. Again, the number of examples is too small to draw general conclusions about the performance of the extended CLR analyzer. Nonetheless, the extension allows HAIKU to adapt to user assignments and move beyond the static marker→CLR mappings and preference rules, making it a potentially significant improvement in CLRA.

2.7 An Example

This section shows the CLRA interactions for three example sentences. In order to illustrate the effects of stored attribute patterns, all three examples have the same connective (*if*) and express the same CLR (Enablement).

(81) *If the end gaps of the piston rings are aligned, oil may leak.*

(82) *It is possible to repair the cylinder with a cylinder hone if scuffing damage is light.*

(83) *If the owner cares for the engine, it will probably serve him for years.*

Since (81) is the first sentence, there are no stored attribute patterns. CLRA holds competitions between the three CLRs in the CLR marker dictionary for *if*: Enablement, Entailment and Prevention (see Figure 1). Enablement wins the competition against

Entailment because of the weak modal (*may*) in E2.[9] Enablement wins the competition against Prevention because the polarity of E2 is positive. Entailment also wins against Prevention because E2 is positive. The system suggests Enablement to the user, who accepts the CLR. The following attribute pattern is stored for future assignments.

if	present_simple/ no_modal	pos	may_present/ weak_modal	pos	enab

```
String (81) If the end gaps of the piston rings are aligned, oil may
            leak.

HAIKU: Clause Level Relationship Analysis of current input ...

There is a Clause-Level Relationship marked by 'if':
  'oil may leak'
     'if'
  'the end gaps of the piston rings are aligned'

CLR competition between enab and entl... enab wins.
CLR competition between enab and prev... enab wins.
CLR competition between entl and prev... entl wins.

Results (maximum is 4):
    enab   entl   prev
  +------+------+------+
      4      2      0

The CLR Analyzer's best suggestion(s) for this input:
  (1)   Enablement (enab)

> Please enter a number between 1 and 1
  or enter a valid CLR for this relationship (a to abort): 1

Your CLR assignment will be stored as:
  'the end gaps of the piston rings are aligned'
     <enables>
  'oil may leak'

> Do you accept this assignment
  (enter r to reverse the arguments, a to abort) [Y/n/r/a]? Y
```

Figure 1: CLRA interaction for (81)

The tense/modality feature for E2 in (82) is `present_simple/no_modal`. The only stored attribute pattern has an E2 tense/modality of `may_present/weak_modal`, which is not a perfect match and cannot be used for (82). HAIKU once more proceeds with CLR

[9] Recall that for the marker *if* the first syntactic argument (main clause) is the second CLR argument (see section 2.4).

competitions for the CLRs marked by *if* (see Figure 2). Entailment wins over Enablement, because there is no modal auxiliary in E2 and HAIKU has no knowledge of modal constructions such as *it is possible....* Enablement cannot be preferred over Prevention since there is no modal in E2. Prevention cannot be preferred over Enablement since the polarity of E2 is positive. So Enablement and Prevention tie. Entailment wins over Prevention since E2 is positive and *if* is a positive connective. The user rejects the system's suggestion of Entailment. The attribute pattern for this assignment will be stored as-is, but since this is a user-supplied CLR, HAIKU will also store the lgg of this pattern and any stored patterns with the same connective and CLR. The pattern from the interaction for (81) qualifies, resulting in the storage of two new attribute patterns.

if	present_simple/ no_modal	pos	may_present/ weak_modal	pos	enab
if	present_simple/ no_modal	pos	*X*	pos	enab

```
String (82) It is possible to repair the cylinder with a cylinder hone
            if scuffing damage is light.

HAIKU: Clause-Level Relationship Analysis of current input ...

There is a Clause-Level Relationship marked by 'if':
  'it is possible to repair the cylinder with a cylinder hone'
     'if'
  'scuffing damage is light'

CLR competition between enab and entl... entl wins.
CLR competition between enab and prev... tie.
CLR competition between entl and prev... entl wins.

Results (maximum is 4):
    enab   entl   prev
  +------+------+------+
      1      4      1

The CLR Analyzer's best suggestion(s) for this input:
  (1)   Entailment (entl)

> Please enter a number between 1 and 1
  or enter a valid CLR for this relationship (a to abort): enab
```

```
Your CLR assignment will be stored as:
  'scuffing damage is light'
     <enables>
  'it is possible to repair the cylinder with a cylinder hone'

> Do you accept this assignment
  (enter r to reverse the arguments, a to abort) [Y/n/r/a]? Y
```

Figure 2: CLRA interaction for (82)

Example (83) also has weaker modality expressed by something other than a modal auxiliary (the adverb *probably*). The tense/modality attribute for E2 in (83) is `future_simple/strong_modal`. The attributes match one of the stored patterns (the lgg of patterns corresponding to (81) and (82)). HAIKU suggests the CLR associated with the matching stored pattern: Enablement.[10] The user accepts the CLR, resulting in the storage of a single attribute pattern only.

if	present_simple/ no_modal	pos	future_simple/ no_modal	pos	enab

```
String (83) If the owner cares for the engine, it will probably serve
            him for years.

HAIKU: Clause-Level Relationship Analysis of current input ...

There is a Clause-Level Relationship marked by 'if':
  'the engine will probably serve the owner for years'
     'if'
  'the owner cares for the engine'

The CLR Analyzer's best suggestion(s) for this input:
  (1)   Enablement (enab)

> Please enter a number between 1 and 1
  or enter a valid CLR for this relationship (a to abort): 1

Your CLR assignment will be stored as:
  'the owner cares for the engine'
     <enables>
  'the engine will probably serve the owner for years'

> Do you accept this assignment
  (enter r to reverse the arguments, a to abort) [Y/n/r/a]? Y
```

Figure 3: CLRA interaction for (83)

[10] Preference rule competitions would have favoured Entailment for (83) because the strong modal and positive polarity in E2.

2.8 Chapter Summary

HAIKU's clause level relationship analyzer assigns CLRs to clauses in coordinate, subordinate and correlative syntactic relationships. CLRA looks up the clausal connective in the CLR marker dictionary to find a subset of candidate CLRs. These candidates compete using preference rules that choose one CLR over another depending on the polarity of the connective and the tense, modality and polarity of the verb phrases in the clauses. The CLR with the most competition points is suggested to the user for approval.

Since the CLR marker dictionary and preference rules do not change during the analysis of a text, HAIKU risks repeating incorrect analyses. To avoid this behaviour, for each CLR assignment the system stores an attribute pattern containing the connective, the syntactic verb phrase features of each clause and the CLR assigned. Patterns that result from user assignments (indicating incorrect system suggestions) are compared to existing stored patterns with the same connective and CLR. Where attribute values are different, a new pattern with variable attribute values is stored in an attempt to isolate relevant attributes. HAIKU consults the stored patterns to find CLRs during future analyses.

Diagnostic evaluation has shown that the marker dictionary and preference rules are accurate when presented with a variety of verb phrase features. Performance evaluation has shown that in complete texts simpler tenses and modalities are more common and that modality is often expressed with adverbs or other constructions instead of modal auxiliaries. Experiments have also underlined CLRA's sensitivity to parse errors, resulting in low coverage of test texts. This low coverage along with the relative infrequency of complex sentences make it difficult to base conclusive claims about performance on these experiments.

The CLR analyzer did not place a large burden on the user for either the *clouds* experiment or the *small engines* experiment: the average onus rating for CLR interactions

over both experiments was 0.07. No new CLRs were needed to cover the relationships in either text.

The CLRA extension to learn from user assignments has so far shown no improvement in the number of correct system CLR assignments, though it does make fewer repeated errors, which are particularly frustrating for the user.

This chapter described the assignment of semantic relationships between clauses within a complex sentence. In the next chapter I go inside the clauses to investigate the relationships between a verb and its syntactic arguments: cases.

3 Case Relationships

Case relationships are assigned to a verb and its positional, prepositional and adverbial arguments. Evaluating lists of cases involves comparing them to other lists and checking their coverage of lexical and syntactic markers. The occurrence of cases in texts can be used as an indicator of their generality and their practical coverage.

This chapter is not about case analysis, the second part of HAIKU's three-part analysis. It is about the case relationships themselves.[1] Of HAIKU's three sets of semantic relationships, the cases are the oldest and have been used in the analysis of more texts than either the clause level relationships or the noun modifier relationships. The processes of

[1] Elements of case analysis inevitably creep into my presentation for coherence (sections 3.2.4, 3.6). Case analysis in HAIKU was a group undertaking in TANKA driven and implemented by Sylvain Delisle (Delisle 1994; Delisle *et al.* 1996). Development of the case list and case marker dictionary and evaluation of the case system were driven by me under the same TANKA wing.

constructing the case set and evaluating it described in detail in this chapter are similar for those other lists.

3.1 Introduction

This introduction gives background on case theory and valency theory, and on other sets of case relationships. The next section (3.2) describes the lexical and syntactic elements that mark cases in clauses. The markers played a leading role in the construction and evaluation of the list of cases (as well as being used in the interactive case analyzer). In section 3.3 I describe the process of designing HAIKU's case system and define criteria for its evaluation. Case definitions appear in 3.4 along with examples. An evaluation of the case system is described in section 3.6.

3.1.1 Cases

Cases represent the semantic relationships expressed syntactically by a verb and its arguments. The verb expresses an event (see section 1.1.1); its arguments express the participants in and circumstances around the event. Researchers building natural language processing systems that use cases (or case-like relationships) seldom devote much effort to developing a set of cases. Often the use of cases is not even explicitly acknowledged. Consider the description from Grishman (1995) of one of the tasks in the popular MUC competition: "The template-filling task for MUC-6 involves the extraction of information about a specified class of events and the filling of a template for each instance of such an event." The templates consist of a set of predefined *template elements* for people, organizations and artifacts involved in the event, *i.e.*, the participants in the event. For example, a *purchasing* activity template consists of roles for the seller, buyer and object purchased. These template elements comprise an application-specific case system, but a case system nonetheless. Even papers on systems that make explicit use of semantic cases (such as Oflazer and Yilmaz 1996) rarely list a complete set of cases and even less often offer any justification for them.

The linguistics community has devoted more attention to the study of case systems (see Somers 1987). These studies, however, are usually driven by linguistic motivations such as, for example, language universals and psychological plausibility. The resulting case systems provide a starting point for case systems intended for computer-based applications such as information extraction or computer aided knowledge acquisition from text.

This chapter describes the construction of a complete case system that brings together elements of existing case systems and evaluates the coverage of the cases on syntactic phenomena and in real texts. The evaluation offers *practical* justifications for the cases, following the lead of Wilks *et al.* (1996) who argue against validating a set of semantic primitives against some external set of entities (such as some unknown ideal set of psychological primitive concepts). They hold that there can be no direct, independent justification for individual primitives, only practical motivations for the inclusion of a given primitive in a set.

3.1.2 Case Theory

Case theory (Fillmore 1968; Somers 1987) focuses on the simple finite clause and on the main verb within it. A case captures that part of the meaning of a syntactic element conveyed by its relationship with the verb. The occurrence of a case in a text is signaled by a case marker. Markers are either lexical (for example, a preposition that introduces a prepositional phrase) or positional (subject, direct object, indirect object). Traditional case theory requires that there be exactly one semantic case assigned to each syntactic argument in a proposition (Fillmore 1968). The restriction can be traced back to case theory's roots in the study of grammatical roles and morphological case (Bruce 1975).

3.1.3 Valency Theory

Valency theory (Somers 1987) deals with the types and number of verb arguments in a clause. The number of required verb arguments depends on the type of complementation of the verb. For example, one sense of the verb *give* is ditransitive and requires a subject and two objects. Cases assigned to required argument positions of a particular verb are

considered *core* roles for that verb, while cases assigned to optional verb argument positions (such as adverbials) are considered *peripheral.*

Some cases (such as Agent and Object) are more often assigned to required verb argument positions and are therefore more frequently core. Others (such as temporal or locative cases) are more often assigned to optional verb argument positions and are therefore more frequently peripheral. It is common for case system designers to distinguish cases as absolutely core or peripheral. But consider example (84).

(84) The meeting lasted [six hours $_{pobj}$*].*

The case assigned to the direct object (written `pobj`—see section 3.2.1) clearly must be a temporal case (such as Duration). Since the direct object is a required argument for the monotransitive use of the verb *last*, the Duration case is core in this example. In contrast, (85) shows a peripheral use of the same case, since it is assigned to an optional prepositional phrase argument.

(85) I read the minutes [for six hours $_{pp}$*].*

Systems that contain core cases only sometimes have a locative case and rarely have a temporal case. To account for sentences such as (85), the temporal nature of the direct object is considered *incorporated* in the verb and a simple Object case is assigned. Alternatively, a verb such as *last* may be considered intransitive with a temporal preposition (such as *for*) lexicalized in the verb (Cook 1989).

For the purposes of automated text analysis the identification of cases that are core for a given verb sense would be a large knowledge engineering task and no existing dictionary offers help. This information would not really be useful in TANKA since we are concerned with collecting actual patterns of cases as they appear in a text: HAIKU's case analyzer captures the cases that are associated with specific arguments for the verbs in a text. Given subcategorization rules for verbs, HAIKU could actually be used to build a dictionary of cases that appear as core and peripheral (see section 5.2.3).

3.1.4 Other Case Systems

There is a long history of work in case theory and research related to cases from many different areas, including linguistics, computer science and cognitive science. Campe (1994) offers a bibliography of over 6,600 references related to case. In this section I will discuss some of the more theoretical work in case theory then some of the computational work in case systems and text analysis. The distinction is not always perfect.

Fillmore (1968) presents a small set of cases that has become the kernel of many subsequent case systems. His cases can be thought of as a set of semantic relationships underlying surface level case, which is expressed by morphology, word order, etc. Fillmore's original list consisted of the six relationships listed in Table 14.

Agentive	Instrumental	Dative
Factitive	Locative	Objective

Table 14: List of cases from Fillmore (1968)

Bruce (1975) explores the relationship between surface case (syntactic level) and deep case (semantic level). He attempts to define a single subjective rule for deciding which cases are "central to an event description"—essentially, which cases are core for certain events. One of his observations is particularly relevant to the TANKA project. He recognizes that it is important to measure the "goodness" of a case system, although he doesn't offer a solution. He does mention that "one can only speak of the goodness of a case system relative to a problem situation" (Bruce 1975: 357) and that different problems require different levels of granularity in case systems (see section 3.3.2). Bruce also surveys several theoretical case systems and several practical case systems in use in early applications. Many of these systems are specific to the domain for which they were created.

Larson (1984) presents a list of cases to be used as an intermediate representation in a machine translation system. The computational setting demands more emphasis on identifying a *complete* set of roles (both core and peripheral) for all verb arguments. Larson's list of twelve cases appears in Table 15.

Agent	Instrument	Beneficiary	Affected
Resultant	Goal	Location	Time
Accompaniment	Causer	Manner	Measure

Table 15: List of cases from Larson (1984)

The theory of conceptual graphs (Sowa 1984) uses a set of *conceptual relations* as semantic links between concepts in a graph. Conceptual relations linking acts with other concepts are borrowed directly from the case relations in case theory: "Arcs of the graphs correspond to the function words and case relations of natural language" (Sowa 1984: 20). Sowa provides a complete list of conceptual relations along with maximally general concepts they may link (according to a concept hierarchy). As is common, relations are defined (at least in part) by example sentences.

	source	***path***	***goal***	***local***
Active	instigator of action	instrument or means	intended result or active recipient	non-passive patient
Objective	original state or material	counter-instrument passive means	result state or factitive	undergoing change-of-state
Dative psychological possession	stimulus original owner	medium medium/price	experiencer recipient	content thing transferred
Locative	place from where	space traversed	final destination	static position
Temporal	time since	duration	time until	time at which
Ambient	reason	manner	aim or consequence	condition

Table 16: The case grid from Somers (1987)

Somers (1987) traces the history of case theory from its roots in valency theory through Fillmore and others to arrive at his own set of case roles. Somers' set of twenty-four cases takes the form of a grid of four columns by six rows (Table 16). The four columns are inspired by the localist division of relations into Source, Path, Goal and Local dimensions (Anderson 1971). Cases appear at the intersection of each row and column. This organization is an attempt to provide a principled basis for the construction of a case

system, as opposed to the arbitrariness of systems based on intuition. The requirement that each category of cases map exactly onto the four dimensions seems at times artificial. Although the localist categories are intuitive, whether there is cognitive support for fitting all cases into such categories is unclear.

Cook (1989) surveys several different models of case theory including his own Matrix Model. His model distinguishes *propositional* cases from *modal* cases. Propositional cases are those essential to a verb's meaning (roughly, core); modal cases are optional adjuncts of any predication (peripheral). Cook's work deals primarily with the propositional cases, though he recognizes the utility of modal cases to automated text analysis. The matrix model contains five propositional cases; Cook also provides a list of modal cases that should appear in any case system that includes modal cases (Table 17). The list of cases was derived from an analysis of the verb-argument relationships in 300 English sentences (Cook 1979) taken from Hemingway. A maximum of three propositional cases may appear in the case frame for a given verb. Object must appear in all case frames and may appear more than once (for stative verbs and object complements, for example). The other cases can appear at most once. A novel aspect of Cook's work is a discussion of criteria for judging the completeness of a case system. These criteria include comparison to existing case systems and extensive use of the cases in text analysis.

PROPOSITIONAL				
Agent	Experiencer	Benefactive	Object	Locative
MODAL				
Time	Manner	Instrument	Cause	
Result	Purpose	outer Locative	outer Benefactive	

Table 17: Propositional and modal cases from Cook (1989)

In work not directly related to the development of case systems, Levin (1993) argues that verbs with similar syntactic behaviour share meaning components. She groups together English verbs that behave similarly with respect to diathesis alternations and attempts to identify elements of meaning common to the verbs in each group. For example, verbs like

cut often have an Agent, an Object and an Instrument. Different alternations of the verb *cut* allow any one of these cases to occupy the syntactic subject position (86).

(86) *[Kettleman $_{subj/Agent}$] cut the bagel.*
[Fresh bagels $_{subj/Object}$] cut easily.
[Sharp knives $_{subj/Instrument}$] cut well.

Levin attempts to identify the semantic features of the verb that permit certain alternations. Her success in this endeavour provides support for the notion that part of a verb's meaning is reflected in the semantic roles allowed to appear in each syntactic argument position. Levin's work also shows the tremendous variation possible in the distribution of case roles in required verb argument positions. This variation shows that the same case can often be assigned to either a required or optional verb argument position, further blurring the distinction between core and peripheral cases.

Van Valin (1993) offers two tiers of semantic roles: *thematic relations* and *macroroles*. Thematic relations correspond to the case roles Agent, Effector, Experiencer, Locative, Theme and Patient. The macroroles Actor and Undergoer serve to rank these thematic relations in the order listed above, with Agent having the highest degree of *Actor-ness* and Patient having the highest degree of *Undergoer-ness* (Table 18). Van Valin's Role and Reference Grammar theory holds that it is this ranking along with verb specific preference rules that allows humans to assign thematic relations to verb arguments.

macroroles	Actor					Undergoer
	←					→
thematic relations	Agent	Effector	Experiencer	Locative	Theme	Patient

Table 18: Macroroles and thematic relations from van Valin (1993)

Sparck Jones & Boguraev (1987) avoid the arbitrariness of case lists based on intuition or assumptions about mental models. Instead, they define a set of cases based on language data. After deciding that their target set should consist of twenty to thirty cases, they studied a substantial list of example sentences illustrating the different senses of the English prepositions. They arrived at their final set of twenty-eight cases by mapping the preposition senses to candidate cases. The resulting cases are defined only in terms of the

example sentences from which they were derived. Without definitions, the distinction between similar cases is not always obvious. Sparck Jones and Boguraev also do not take into consideration the different syntactic functions of prepositions. Since prepositions do not always mark relationships between verbs and arguments, their set of cases includes several non-case semantic relationships. For example, the Possessed-By relationship can only link nominals; the State relationship is included for predicating adjectives. Finally, there is no evaluation of the level of generality of each case: some of the cases are very specific (such as Force) while there is a single undiscriminated Location case to account for all locative relationships.

Slator *et al.* (1990) derive semantic classes based on a study of the twenty-five most common English prepositions. 184 different senses of the prepositions are clustered into 46 classes based on a semi-automatically computed measure of similarity of their (conventional) dictionary definitions. The classes are then given names according to human reflection on the shared semantic properties of the senses in each cluster. What is striking in the results is the similarity between these empirically derived classes and the traditional cases found in more arbitrary, intuition-based case systems. The classes that do not map straightforwardly onto common case roles often express non-case uses of the prepositions. These classes include relationships such as Attribute and Comparison. The relatively large number of classes can be attributed to these non-case classes and the small number of preposition senses per class (on average). The classes are correspondingly fine grained, with as many as fifteen classes that appear to cover locative relationships. Nonetheless, the similarity between these purely empirically derived classes and case roles common to other systems lends empirical support to the kinds of semantic relationships intuited by case researchers.

Velardi *et al.* (1991) attack the problem of acquiring shallow semantic knowledge from text semi-automatically. Their system uses a set of mapping rules between *syntactic collocates* and semantic relations. The syntactic collocates consist of pairs of syntactically connected constituents, such as subject-verb pairs, verb-object pairs, adjective-noun pairs and prepositional phrases that postmodify nouns. The output is a set of concept-relation-

concept triples, where concepts are generalizations of individual words based on a small, domain dependent, hand-coded concept hierarchy. The list of potential relations for a given syntactic collocate is narrowed by selectional restrictions between general concepts in the hierarchy. Finally, a user can accept or reject analyses for inclusion in a permanent knowledge base. A complete list of semantic relations is not given, though the output is in the Conceptual Graph notation (Sowa 1984) and appears to use common Conceptual Graph relations.

Wu (1993) restricts thematic role frames (case patterns) to a maximum of four thematic roles per frame, and relies on the nesting of role frames to account for verbs with more than four arguments. Nesting frames allows the system to assign only core roles in the top level frame while relegating peripheral roles to nested frames. According to Wu, a side effect of using nested structures is that "surface cases do not map straightforwardly onto thematic roles in a one-to-one fashion" (Wu 1993: 327). In a system attempting to represent the semantics of a text in a form that closely matches its surface syntax, a one-to-one mapping between surface cases and semantic roles is desirable. Systems whose semantic representation does not map easily to surface syntax become much more dependent on the amount and form of precoded semantic background knowledge.

Pustejovsky & Busa (1995) present the details of the temporal template elements for the MUC competition. The four 'time objects' During, Before, After and On are equivalent to Spark Jones & Boguraev's (1987) temporal cases, and also map roughly to the localist Path, Source, Goal and Local categories. Although Pustejovsky and Busa do not use the term *cases*, they show a mapping from prepositional (and other lexical) markers to the four time objects. For example, *in*, *throughout*, *for*, *late*, *early*, *beginning* of and *end of* mark the During object; the Before object is marked by *before*, *ending*, *until* and *by*.

3.2 Case Markers

A case marker is a syntactic or lexical indicator of a case in a sentence. HAIKU recognizes three types of case marker: positional, prepositional and adverbial.

3.2.1 Positional Markers

Positional markers mark a case by their syntactic position in the clause. They are the syntactic subject, direct object and indirect object of the verb. In this dissertation I use the symbols `psubj`, `pobj` and `piobj`, which refer to *positional subject, positional direct object* and *positional indirect object*. These three syntactic positions often mark cases such as Agent, Beneficiary, Experiencer, Object and Recipient. The constituents in these positions fill the cases marked by the position itself.

3.2.2 Prepositional Markers

A prepositional phrase consists of a preposition and a noun phrase complement. Prepositions express a relationship between their complement and some other constituent in a sentence (see Quirk *et al.* 1985: 9.1). A preposition expressing a relationship between its complement and a verb marks a case. The prepositional phrase is *attached to* the verb and the prepositional complement is the case filler. A preposition expressing a relationship between its complement and a noun phrase (as opposed to a verb) does not mark a case. That preposition is attached to a noun and marks a noun modifier relationship (see section 4.5).

In example (87), the prepositional phrase *on the printer* is attached to the verb *printed*. The noun phrase *the printer* fills that verb's Instrument case. In (88), the prepositional phrase *on the printer on his desk* is also attached to the verb *printed*. The noun phrase *the printer on his desk* fills the Instrument case. The prepositional phrase *on his desk*, however, postmodifies the noun *printer* and fills a noun modifier relationship of the noun *printer*, not a case of the verb *printed*.

(87) Bonnie printed the paper [on [the printer $_{np}$*]* $_{pp}$*].*

(88) Clyde printed the paper [on [the printer [on his desk $_{pp}$*]* $_{np}$*]* $_{pp}$*].*

Because certain prepositions and conjunctions are homographic, conjunctions may seem to mark cases. In (89), the conjunction *after* appears to mark TimeAt.

(89) Fred came [after $_{conj}$*] Barney left.*

A conjunction joining clauses does not mark a case however because it does not signal a relationship between a verb and one of its arguments. Instead, it marks a clause level relationship.

3.2.3 Adverbial Markers

Adverbials are the third type of case marker. The same semantic relationships that are expressed by prepositional markers can also often be marked by adverbials. In (90), *at* marks and *the same time* fills the TimeAt case; in the paraphrase in (91), *simultaneously* expresses the same case.

(90) The two events occurred [at the same time $_{pp}$*].*

(91) The two events occurred [simultaneously $_{adv}$*].*

Based on the similar distribution of prepositional phrases and adverbials in denoting the circumstances of events, adverbials are valid case markers in HAIKU. An adverbial case marker both marks a case and fills it.

3.2.4 Marker Order

For case analysis of a given clause, HAIKU builds an ordered pattern of markers for the main verb. In the pattern, positional arguments appear as the symbols `psubj`, `pobj` and `piobj`; prepositional phrase arguments are represented by their prepositional marker; adverbial arguments are represented by the symbol `adv`.

`psubj` > `pobj` > `piobj` > *preps* > `adv`

This ordering reflects the unmarked SVO (subject-verb-object) word order in English as well as the notion of clause structure element centrality (Quirk et al. 1985: 2.13). The notion of centrality comes from the observation that the verb is the most central element of the clause; the adverbial is the least central, or most peripheral. Between these two extremes, the subject, then the objects and then any other non-adverbial complements (such as prepositional phrases) are decreasingly central. Clause element centrality has an obvious relationship to the distinction between core and peripheral cases. Those cases

assigned to markers in more central positions would more likely be considered core roles for a verb.

3.3 Case System Design

In this section I describe the process of defining HAIKU's cases. The process, though guided by English marker data, example sentences and test texts, is not purely data-driven: the initial case set was inspired by cases from existing systems. Nonetheless, the role of language data in the evaluation of the cases grounds the set in linguistic fact, not just in intuition.

The first step in the process was to find a rough union of others' case systems (such as those described in section 3.1.4). Any case appearing in any other list was a candidate for inclusion in HAIKU's set. Only one case was selected among every strongly similar group of cases, even if they had different names in different systems. For example, HAIKU has a single Object case to account for cases variously named Affected, Object, Patient and ThingTransferred. Also ignored are cases not representing relationships between verbs and their arguments, such as the Attribute and Possessed-By cases in Sparck Jones and Boguraev (1987), since those non-case relationships are covered by the clause level relationships or the noun modifier relationships.

The second step in construction was to build a comprehensive list of case markers and map them to the cases. The set of cases was adjusted according to the extent to which cases were represented by the markers in that list. The distribution of data in the marker list is not the same as in complete texts. In this sense, using the case marker dictionary to test the coverage of the cases was more a *diagnostic* evaluation of the case set. The evaluation in section 3.6 represents a *performance* evaluation. Criteria for both evaluations are discussed in section 3.3.2.

3.3.1 The Case Marker Dictionary

The case marker dictionary was built by searching electronic and conventional dictionaries[2] for preposition and adverb senses. The positional case markers (`psubj`, `pobj` and `piobj`) were also included in the case marker dictionary.

For the prepositions each distinct sense was examined to decide which cases were appropriate to that sense. If a sense corresponded to a case, an entry mapping the preposition to that case was added to the case marker dictionary. If no case adequately captured a particular marker usage, the set of cases was reexamined to determine if a new case was needed or if the scope of an existing case should be redefined. At the end of the process any case that was weakly represented in the marker list was reexamined to see if it was necessary.

The source dictionaries were also consulted to find adverbial cases markers, such as those in (92) and (93).

(92) *This printer prints [quickly* $_{adv/Manner}$*].*

(93) *We backup the hard disk [daily* $_{adv/Frequency}$*].*

Adding mappings between all adverb case markers and cases would not be desirable since the class of adverbs is not closed. An adverb was only added to the case marker list if it had a sense corresponding to a case other than Manner, the most common case marked by adverbs. If the adverb also marked Manner, a mapping between the adverb and Manner was included in the list. Any adverb not in the list is assumed to mark Manner only. Conversely, adverbs that *do* appear in the list are taken to mark Manner only if there is an explicit entry mapping them to Manner.

Every marker→case mapping in the marker list includes an example sentence (see Appendix II for a sample of the entries for prepositions). In addition to providing the user

---

[2] The dictionaries I used most were the *COBUILD* (Sinclair 1991), the unabridged *Random House* (Stein 1983) and Grady Ward's *Moby* lists. I also consulted *LDOCE* (Summers 1987) and *Fowler's Modern English Usage* (Fowler 1984) to confirm and clarify marker senses.

Implementation Note

The case marker dictionary is compiled with HAIKU and there is no facility to load new mappings at run time. Nonetheless, adding new entries and new mappings to existing entries is a simple matter of adding Prolog facts to the dictionary in the form shown in Appendix II. Since HAIKU saves all marker→case assignments in a separate output file, new mappings resulting from a HAIKU session can simply be copied over to the case marker dictionary.

with an illustration of a given marker→case pair (during case analysis), the example sentences helped tune the case set. An example sentence was sometimes a compelling argument to include a case. Conversely, cases for which an example sentence could not easily be invented were considered dubious. Another technique to tweak the case set was to try to construct an example sentence containing two similar cases. When it was not possible to compose a sentence containing both cases, they might be merged into a single more general one.

3.3.2 Evaluation Criteria

I now identify three desirable characteristics of a case system. These are certainly not the only possible criteria for evaluating cases, but they have proven to be useful for finding weaknesses in the way the case set covers empirical data (both exhaustively enumerated data, such as the case marker dictionary, and data from real texts).

Generality

Cases must generalize to some degree the role of arguments of a verb. Individual verb-argument pairs themselves could not be used as cases, since they would be nothing more than a unique label for the relationship and would add nothing to its interpretation. For example, a BreakWindow case assigned to the direct object in (94) would not add any information to the interpretation of that sentence; and it would not account for the similarity of the roles played by the direct objects in (94), (95) and (96).

(94) *Alphonse broke [the window* $_{pobj}$*]*

(95) *Alphonse broke [the pane* $_{pobj}$*]*

(96) *Alphonse shattered [the window* $_{pobj}$*]*

At the other extreme, verb-argument pairs could be represented by just two roles corresponding to *participants* (P) and *circumstances* (C). This trivial case system would probably be too general for any natural language processing task since the designation of verb arguments as participants or circumstances is no more informative than the syntactic analysis of verb arguments as objects or adverbials.

(97) [Gaston $_P$] broke [the window $_P$] [yesterday $_C$] with [a rock $_P$].

The two extremes inspire the first criterion for the construction of practical case systems. Cases should be specific enough to distinguish between the roles of each verb argument in a given clause. A Participant case is too general to account for the different roles played by the subject, direct object and prepositional phrase in (97). On the other hand, cases should generalize the semantic relationships in a clause. That is, each case must account for relationships between distinct verb-argument pairs from different clauses. The BreakWindow case would account for only *one* verb-argument pair.

Together the specificity and generality required of a case system will determine how many cases it contains and how closely definitions of the cases match the semantics of the specific verb-argument relationships they generalize. Choice of a particular point between the two extremes on the scale will depend on such things as the application using the case system, the target representation of case analysis and the type of text. For example, the Participant role that I claimed was too general for (97) may be specific enough if the application is a template filling task to determine the entities involved in an act. If the target semantic representation to be produced from case analysis does not make subtle distinctions between semantic roles, then a fine grained case set would be unnecessary. Certain types of texts may also dictate the required generality of a case system. The analysis of geographical texts, for example, may require fine distinctions between a number of locative cases.

Commonly, the overall level of generality of cases is not explicitly addressed by case system designers. There are ways, however, to measure the generality of an existing case set. One measure is the number of times each case is assigned in the analysis of texts.

Although this measure would be valuable in determining the practical generality of a set of cases, it would require that a large amount of text be analyzed to draw any significant conclusions. It also assumes that frequency of occurrence is directly related to generality. Although frequency may hint at which cases are more general than others, it is possible that certain quite generally defined cases are less common in texts than some cases with very restricted, specific definitions. Furthermore, the measure would change as more texts were analyzed, though more analysis would also increase confidence in its value.

A second measure is the number of syntactic phenomena (*i.e.*, markers) accounted for by each case. Although this measure tells us less about the applicability of each case to the analysis of texts, it is related to the first measure of a case's generality: the generality of a case using the first measure is bound from above by the frequency with which that case's markers occur in text. So a case with more markers or commonly occurring markers (such as the positional markers) will potentially account for more verb-argument relationships in a text than a case with fewer or less common markers. This second measure is easily calculated since the markers are usually fixed independent of any texts to be analyzed.

In Barker (1996) I investigate the degree to which each of HAIKU's cases is represented by markers in the case marker dictionary as well as in the *clouds* text. The report also discusses possible adjustments to the case set for cases that have marker representation above or below average. The main results from the investigation appear in section 3.6.2, where they are extended with results from the *small engines* experiment.

Completeness

A set of cases should cover any possible verb-argument relationship. In practice a case system can never be shown to be perfectly complete; conversely, a single counterexample can show that it is incomplete. One measure of the degree to which a given case system is complete is the number of sentences in a text containing relationships covered by the case system. As more clauses are case analyzed within the confines of a given set of cases, confidence in the completeness of that set increases.

A potential pitfall of trying to satisfy the completeness requirement is the proliferation of new cases. If new cases are continually introduced to remedy incompleteness, their number may grow unchecked. Generality, however, ensures that overly specific cases are not added to cover a single verb-argument relationship. A solution is to expand the definition of an existing case to account for a new relationship. The new definition should cover all of the relationships covered by the old definition along with the new relationship, and no others. Documentation of each case's coverage is therefore particularly important.

Uniqueness

A case system should contain no superfluous or redundant cases. A case is superfluous if the relationship it describes never appears in any text between a verb and argument. A case is redundant if every instance of it is also an instance of some other case in the set.

Empirically, a case can never be shown to be absolutely superfluous or redundant: the fact that a case has covered none of the encountered verb-argument pairs in texts does not imply that it will never do so; the fact that the set of encountered relationships covered by a case is a subset of the set of relationships covered by some other case does not imply that the two sets will never diverge.

Nonetheless, redundant and superfluous cases can be avoided by requiring that no case be added to the set arbitrarily. That is, inclusion in the case set must be restricted to cases that cover at least one verb-argument relationship not covered by any other case in the set. Redundancy can also be avoided by defining explicitly the semantic relationships covered by a case and by contrasting these with the definitions of other cases.

3.3.3 Using the Criteria to Guide Case Selection

The initial grain of HAIKU's case system was suggested by that of the existing systems from which cases were taken. The case markers further guided adjustment to the specificity of the system. If a single case was marked by several different markers with differing senses, the case was split into more specific cases. Similar cases marked by only

one or two markers were merged into a single more general case. The temporal roles provide examples of both kinds of adjustment.

Many older case systems have only one or two temporal cases. More recent work distinguishes between multiple temporal cases, often corresponding to the four localist categories. HAIKU's cases include Frequency in addition to the common TimeAt, Duration, Before and After, inspired by the prevalence of adverbial case markers for such a role: *hourly*, *daily*, *monthly*, etc. Frequency is clearly a temporal role not captured by TimeAt, Duration, Before or After.

The Before and After cases were richly represented by several different markers, as evidenced also in Pustejovsky & Busa (1995). For example, Before is marked by the prepositions *before*, *by*, *ending*, *till*, *to*, *towards*, *until* and *up to*. With the exception of *before* and possibly *by*, these markers suggest a slightly more specific role describing the ending point of an act of duration. To capture this distinction we split the Before case into TimeBefore and TimeTo. Similarly, we split the After case into TimeAfter and TimeFrom, bringing the number of temporal cases up to seven.

With the introduction of TimeTo and TimeFrom cases, TimeBefore and TimeAfter became very weakly represented by the markers. Only *before* exclusively marked TimeBefore and only *after* exclusively marked TimeAfter. We deleted these two cases, leaving their examples to be covered by the more general TimeAt case.

3.4 The Cases

The resulting set of cases (Table 19) was then compared against those in the systems described in section 3.1.4 to ensure that it had at least as much coverage as those systems.

The cases appear in five groups. The PARTICIPANT group consists of cases in which the argument is usually directly involved in the event expressed by the verb. The CAUSALITY group includes relationships enabling or opposing the event. SPACE and TIME group

PARTICIPANT	
Accompaniment (ACMP)	Experiencer (EXPR)
Agent (AGT)	Instrument (INST)
Beneficiary (BENF)	Object (OBJ)
Exclusion (EXCL)	Recipient (RECP)
CAUSALITY	
Cause (CAUS)	Opposition (OPP)
Effect (EFF)	Purpose (PURP)
SPACE	
Direction (DIR)	LocationThrough (LTRU)
LocationAt (LAT)	LocationTo (LTO)
LocationFrom (LFRM)	Orientation (ORNT)
TIME	
Frequency (FREQ)	TimeThrough (TTRU)
TimeAt (TAT)	TimeTo (TTO)
TimeFrom (TFRM)	
QUALITY	
Content (CONT)	Measure (MEAS)
Manner (MANR)	Order (ORD)
Material (MATR)	

Table 19: The cases (with abbreviations)

relationships that place the event at an absolute or relative position in space or time. Finally, QUALITY groups the remaining cases that represent various other relationships between a verb and its arguments.

The groups are more useful when talking about the cases than when using them. For example, the division into groups shows clearly the parallel between each of the temporal cases and their spatial analogs. Moreover, since each group contains cases that share some property, clashes or overlaps between cases are much more likely to occur within groups. For example, it is more likely that Effect and Purpose will need to be distinguished explicitly than Purpose and LocationTo; TimeThrough is more likely redundant because its instances are covered by TimeAt than because they are covered by Agent.

3.4.1 Case Glossary

This section describes the meaning and coverage of each case. Each description is followed by one or more example sentences chosen to reflect typical usage or to illustrate the scope of each case.

Participant

Accompaniment (ACMP)

The Accompaniment case represents one or more entities accompanying another entity (usually the Agent, Experiencer or Object) involved in an event. Examples (98), (99) and (100) show accompaniment of Agent, Experiencer and Object.

(98) [I $_{AGT}$] eat supper with [my family $_{ACMP}$].

(99) [I $_{EXPR}$] live with [my family $_{ACMP}$].

(100) I eat [my peas $_{OBJ}$] with [honey $_{ACMP}$].

Agent (AGT)

The Agent case represents the initiator or performer of an event. An Agent is typically a sentient being or some entity treated as sentient to some degree within the domain. This case differs from the Experiencer case in that the Agent intentionally performs an act or actively participates in an event. The Agent is expressed by the syntactic subject of the verb or as the object of the preposition *by* in passive clauses.

(101) [The database manager $_{psubj/AGT}$] retrieved the records.

Example (101) shows an Agent, *The database manager*. Although database manager modules are not usually thought of as sentient, in the domain of Computer Science they often perform complex tasks. In the absence of a truly sentient initiator, other entities (such as database managers) can be Agents. In (102) the Agent would be *The analyst* and

Implementation Note

In practice, for any particular assignment the choice of case depends on the reader's interpretation of the sentence, especially for more abstract examples. HAIKU allows the user to invent new cases if finer grain is required for a text. These user-defined cases are stored in a separate file that can be loaded at the beginning of a new session. When assigning a case in a sentence the user can choose one from the original set or from the user-defined cases.

the *database manager* fills the role of Instrument.

(102) [The analyst $_{AGT}$] retrieved the records with [the database manager $_{INST}$].

Beneficiary (BENF)

The Beneficiary case represents the entity that benefits from the state resulting from the event. The state may be to the Beneficiary's advantage or disadvantage, and it may be intentional or accidental. Typically the Beneficiary is an animate being or an organization. It may correspond to the syntactic indirect object of the verb if the indirect object is not the Recipient of the object of the event.

(103) I wrote [Smilla $_{piobj/BENF}$] a reference letter [to prospective employers $_{pp/RECP}$].

(104) I wrote [Smilla $_{piobj/BENF}$] [a reference letter [to prospective employers $_{pp}$] $_{pobj/OBJ}$].

(105) This year's rains produced a bumper crop [for the farmer $_{pp/BENF}$].

Example (103) shows the indirect object (*Smilla*) filling the Beneficiary case with *prospective employers* filling the Recipient case. In (104) there is no verb argument to fill the Recipient case (*to prospective employers* is attached to *letter*, not *wrote*). Nonetheless, since the writing was done to the advantage (or disadvantage) of *Smilla*, Beneficiary is again an appropriate case. Example (105) illustrates the fact that the filler of the Beneficiary case need not be the *intended* beneficiary of the event. Clearly, the production of a bumper crop for the farmer's benefit was not intentional.

Exclusion (EXCL)

The Exclusion case represents an entity *not* included in a group or not accompanying another entity or entities in an event or state. It can also represent the entity that substitutes for another whose involvement in the event is expected, or whose lack of involvement is significant.

(106) Everybody slept except [Jeff $_{EXCL}$].

(107) Jeff played instead of [Howard $_{EXCL}$].

Experiencer (EXPR)

The Experiencer case represents the entity experiencing a state or a sensation. Unlike an Agent, an Experiencer does not intentionally perform an act or actively participate in an event. The Experiencer is typically a sentient being or some entity treated as sentient. It corresponds to the syntactic subject of the verb.

(108) [Ajax $_{psubj/EXPR}$] is sleeping.

Instrument (INST)

The Instrument case represents an entity that is applied or employed in an event. The Instrument for events of transfer is often the medium of the transfer.

(109) He broke the window with [a brick $_{INST}$].

(110) The system administrator notified the users via [email $_{INST}$].

In (109) *a brick* is the entity applied to accomplish the event of breaking the window. In (110) *email* is the medium of transfer of notifying the users.

Object (OBJ)

The entity directly acted upon by the event fills the Object case. The Object often corresponds to the syntactic direct object of the verb but may also be marked by syntactic subject or by a preposition.

(111) Ethel printed [the file $_{pobj/OBJ}$].

(112) [The window $_{psubj/OBJ}$] broke.

(113) They stripped him [of his pride $_{pp/OBJ}$].

Example (111) has the syntactic direct object (*the file*) filling the Object case; (112) shows the less common situation of the syntactic subject (*the window*) filling this case; (113) is an example of the rare occasion where the Object case is marked by the

preposition *of*.[3] *His pride* is the entity that has been stripped while *him* is the location from which it was stripped.[4]

Recipient (RECP)

The Recipient case represents the entity that directly receives the object of an event. The Recipient must be distinguished from the closely related Beneficiary case. Whereas the Beneficiary benefits from the realization of an event, the Recipient takes possession of the event's object. Recipient frequently appears with events describing dative relationships and often corresponds to the syntactic indirect object of the verb.

(114) I sent [the prospective employers $_{piobj/RECP}$*] a reference letter for Smilla.*

(115) I wrote [Smilla $_{piobj/BENF}$*] a reference letter [to prospective employers* $_{pp/RECP}$*].*

In (114) the indirect object of the verb fills the Recipient case. In (115) the indirect object fills the Beneficiary case while the Recipient case is marked by the preposition *to* (see the definition of Beneficiary).

In sentences that contain both cases, Beneficiary is more often marked by the preposition *for* while Recipient is more often marked by the preposition *to*.

Causality

Cause (CAUS)

The Cause is the state or event that makes another event take place. The Cause case may also represent an environment that allows an event to be performed or a state to exist.

(116) He died of [thirst $_{CAUS}$*].*

(117) We acted on [his advice $_{CAUS}$*].*

---

[3] Object marked by *of* is rare but not merely idiomatic. Similar examples using different verbs support the decision to treat *of* as a valid marker for Object: *The deprived him [of his rights* $_{pp/OBJ}$*]*; *They partake [of the bread* $_{pp/OBJ}$*]*.

[4] Example (113) has a person as the abstract location of a feeling. An alternative to allowing both abstract and concrete locations would be to refine each locative case into two separate cases distinguished by the feature ±concrete.

Effect (EFF)

The Effect case represents a state that is the outcome of an event or the result of another state.

(118) The battle will end in [death $_{EFF}$].

Opposition (OPP)

The Opposition case represents an entity that contrasts with or opposes the event but is insufficient to prevent it from happening.

(119) Despite [my warning $_{OPP}$] they persisted.

Purpose (PURP)

The Purpose case represents the state intended to result from the event. This case implies initiation of the event by a sentient being. Purpose differs from Effect in that Purpose is the intended though not necessarily occurring result whereas Effect is the occurring though not necessarily intended result.

(120) The drug was invented for [pain relief $_{PURP}$].

Space

Direction (DIR), LocationFrom (LFRM), LocationThrough (LTRU), LocationTo (LTO)

The spatial cases Direction, LocationFrom, LocationTo and LocationThrough represent positions in some (possibly non-physical) space. To distinguish between them consider an event of motion to be an arrow. The tail of the arrow represents the starting point of the event (LFRM). The head represents the destination of the event (LTO). The direction of the arrow corresponds to the event's bearing in space (DIR). The shaft joining the tail to the head corresponds to the space through which the event passes or extends (LTRU).[5]

(121) I drove [east $_{DIR}$] on [autoroute 640 $_{LTRU}$] from [Laval $_{LFRM}$] to [Repentigny $_{LTO}$].

(122) Look inside [yourself $_{DIR}$] for the answer.

---

[5] In the localist framework, LocationFrom, LocationThrough, LocationTo and LocationAt correspond to Source, Path, Goal and Local. There is no direct analogue for Direction.

(123) They stripped [him $_{LFRM}$] of his pride.

(124) Strange images flitted through [my mind $_{LTRU}$].

(125) His life is going [nowhere $_{LTO}$].

Example (121) shows all four cases together in concrete (physical) usage. Examples (122) to (125) illustrate abstract (non-physical) uses of Direction (*yourself*), LocationFrom (*him*), LocationThrough (*my mind*) and the LocationTo case (*nowhere*).

LocationAt (LAT)

The LocationAt case represents the space in or at which an event occurs or a state exists. This case applies to non-motion events that occur statically in space, or to specific points along the path of a motion event. There is no restriction on the physical extent of a point that fills the LocationAt case. As with other locative cases, LocationAt may be interpreted abstractly (non-physically), and either absolutely or relative to another location.

(126) We stopped at [a restaurant $_{LAT}$] on our way home.

(127) Memory of the accident was hidden in [his subconscious $_{LAT}$].

In (126), a restaurant is a specific point along the path of motion; (127) illustrates an abstract usage of the LocationAt case.

Orientation (ORNT)

The Orientation case represents the bearing an entity involved in an event in terms of the axes on whose origin the object is centred. Less strictly, Orientation may represent the position of parts of an object with respect to each other.

(128) The tower lay on [its side $_{ORNT}$].

(129) The tree stood [erect $_{ORNT}$] despite the heavy ice.

Time

Frequency (FREQ)

The Frequency case represents the rate at which an event or state recurs.

(130) Buses arrive [in five minute intervals $_{pp/FREQ}$*].*

(131) Jennifer washes her truck [daily $_{adv/FREQ}$*].*

The two example sentences show the Frequency case being marked first by a preposition and then by an adverb.

TimeAt (TAT)

The TimeAt case represents the time at which an event takes place or a state exists. It is the temporal analog of the spatial case LocationAt. There is no restriction on the extent of the time unit filling the TimeAt case—the filler need not refer to a measurable instant in time. It may also be another event indicating when the event took place.

(132) He traveled extensively [last year $_{TAT}$*].*

(133) He made his fortune during [his visit to Europe $_{TAT}$*].*

The case filler in (132) is a time unit with measurable extent. In (133) the nominalized event *his visit to Europe* indicates the time when the event (making his fortune) occurred.

TimeFrom (TFRM)

TimeFrom is the case that represents the time of the beginning of an event. It may also represent the moment at which a state began to exist or will begin to exist.

(134) He has been blind since [the war $_{TFRM}$*].*

(135) I will stay at your house from [Tuesday $_{TFRM}$*] to Saturday.*

In (134) *the war* marks the time at which the state began. *He* has been blind from the war until now. In (135), the moment at which the state of staying at your house will begin is in the future (*Tuesday*).

TimeThrough (TTRU)

The TimeThrough case represents the duration of an event or a state. In contrast to TimeAt, the filler of this case must have extent.

(136) The meeting lasted for [six hours $_{TTRU}$*].*

TimeTo (TTO)

The TimeTo case represents the time at which an event of certain duration ended or will end. It may also represent the moment at which a state ceased to exist or will cease to exist.

(137) We worked from nine to [five $_{TTO}$*].*

(138) I will stay at your house until [Tuesday $_{TTO}$*].*

Quality

Content (CONT)

The Content case represents the subject of any type of communication or consideration event. It may also represent the physical filling of a container involved in an event.

(139) John wrote about [volcanoes $_{CONT}$*].*

(140) Albert is concerned about [the economy $_{CONT}$*].*

(141) He filled the container with [milk $_{CONT}$*].*

Manner (MANR)

The Manner case represents a way in which an event is performed or takes place. This case accounts for many common relationships between a verb and its arguments that describe qualitative characteristics of the event or state. Manner is often used in the absence of a more suitable case and can thus be considered a default. This is particularly true for adverbial markers.

(142) Emily writes [with style $_{pp/MANR}$*].*

(143) Michael played [beautifully $_{adv/MANR}$*].*

Material (MATR)

The Material case represents the physical substance composing an entity involved in an event.

(144) We build houses with [brick $_{MATR}$*].*

Measure (MEAS)

The Measure case represents some quantitative property describing the extent of an event or the amount, number or value (including economic worth) of an entity involved in the event. An instance of Measure implies a scale of measurement.

(145) Secretariat won by [three lengths $_{MEAS}$*].*

(146) I bought the car for [five hundred dollars $_{MEAS}$*].*

Order (ORD)

The Order case represents the placement of an entity at relative position within a sequence, or within another structured arrangement of entities participating in an event.

(147) He filed the Baker file before [the Abel file $_{ORD}$*].*

3.5 Assigning Cases

The case analyzer in HAIKU extracts all case markers in a given clause based on their syntactic labels as identified by DIPETT. It assembles these markers into a case marker pattern (CMP). HAIKU then attempts to determine the corresponding case pattern (CP) by comparing the current CMP and verb to CMPs and verbs stored from previous clauses. The technique is reminiscent of case-based reasoning, with previous CMPs and verbs comprising the base of cases.[6]

If the current case marker pattern CMP_c has already appeared in some previous clause $Clause_p$, then the previous case pattern CP_p is suggested to the user for the current clause. If CMP_c has never been encountered, a partial matching algorithm finds the most similar previous marker pattern CMP_s among previous clauses. The measure of CMP distance for the similarity calculation is defined in Delisle *et al.* (1993). Case patterns previously assigned to CMP_s contain candidate cases for the current CP.

---

[6] I will not make any other references to case-based reasoning; the term *case* is already overloaded.

If the system cannot find good candidate cases to suggest to the user (for example when processing the first sentence of a text), the user must supply a case pattern using the definitions of the cases (section 3.4.1) to guide case selection. Once the user has accepted a suggested CP or supplied one, the CP is stored along with CMP_c and the current verb V_c. These stored patterns are accumulated to help in the processing of subsequent clauses.

The implicit assumption in using previous case analyses to guide assignment is that the cases are generalizations that can apply to different verb-marker pairs. The number of times previous case patterns suggested to the user are accepted testifies indirectly to the generalizing power of the cases themselves.

3.6 Evaluation

I have so far described the construction of a set of cases driven by English case marker data from dictionaries and word lists. Evaluation of the set was accomplished by using HAIKU in the case analysis of sample texts.

3.6.1 Case Analyzer Evaluation

I claimed in section 3.3.2 that the evaluation of a case system includes its coverage of clauses from real texts. This section briefly presents the results of case analysis of the *clouds* text and the *small engines* text.

The *clouds* text contains 512 sentences from which DIPETT parsed 439 finite, non-stative clauses usable for case analysis. The system assembled automatically the correct CMP for 69% of the clauses. The user supplied the correct CMP for the remaining clauses.

Starting with an empty processing history, the system suggested zero or more case patterns for each CMP, depending on previous processing as described above. Over all 439 clauses, the system made on average 4.5 CP suggestions per CMP. The maximum number of CP suggestions made for any single clause was fourteen. The increase in this maximum was quick for the first half of the experiment (jumping from zero to eleven

over the first 200 clauses) but slowed for the second half, where the maximum increased by only three over the last 200 clauses.

The correct case pattern was among the system's suggestions for 62% of the 439 clauses—50% for the first 208 clauses and 72% for the next 231 clauses. Figure 4 shows the number of clauses for which the correct pattern was among the system's suggestions (*system*) contrasted with the number that had to be supplied by the user (*user*).

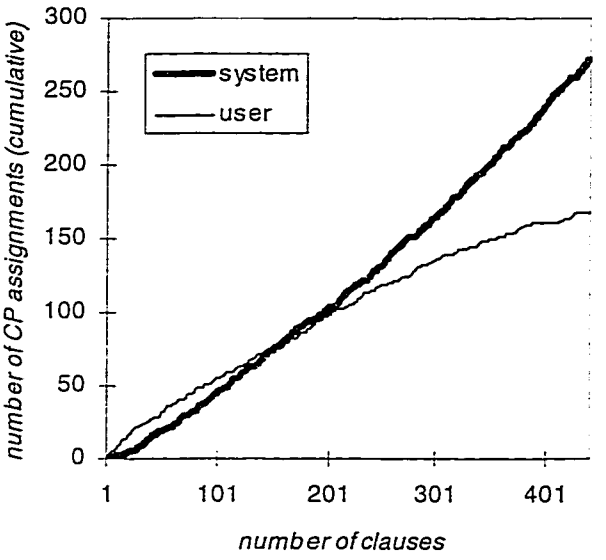


Figure 4: CP assignments in the clouds *experiment*

In the *small engines* experiment, 584 clauses received case analysis. The system made on average 4.4 suggestions for clauses for which it could make any suggestions at all. After fewer than a hundred clauses, the number of system assignments overtook the user assignments for good. By the end of the experiment, the system had suggested a correct case pattern for 66% of the assignments. Figure 5 plots the cumulative number of assignments made by the user against the cumulative number of correct assignments among the system's suggestions. The similarity of the trends in Figure 4 and Figure 5 supports the contention that the case analyzer can learn to make better suggestions to the user.

In order to evaluate the proportion of clauses in the text that received a case analysis, I sampled 100 verbs at random from the *small engines* text, determined their case marker pattern manually and found that 97 of the sampled verbs and their correct CPs appeared in HAIKU's output. With 95% confidence HAIKU extracted between 91.5% and 98.9% of the case patterns in the text. This high coverage is due to the fact that the user is given the opportunity to correct case marker patterns that are incorrect due to misparses.

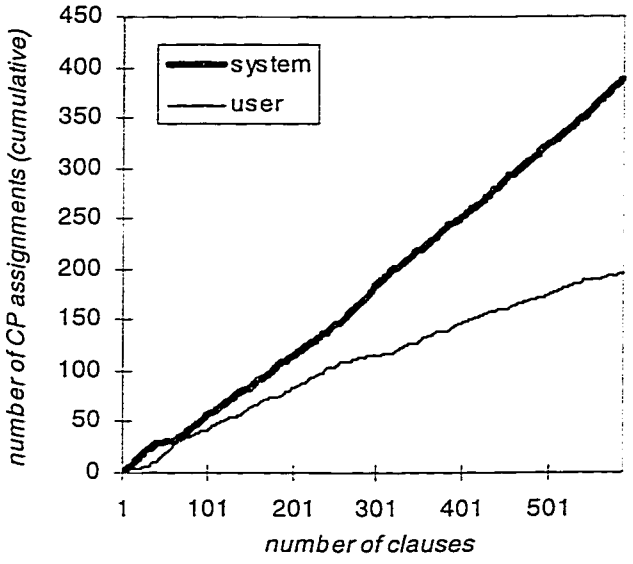


Figure 5: CP assignments in the small engines *experiment*

Finally, for both the *clouds* and *small engines* experiments, a user onus rating was recorded for each user interaction (see section 1.4.4). For the *clouds* experiment user onus was 0 (lowest) for 384 interactions (87%). For 50 interactions (11%) the onus was somewhat higher (1) and for only five interactions (1%) the onus was rated 2. No interactions were rated onus level 3. In the *small engines* experiment the corresponding onus ratings were: 480 onus 0 interactions (82%), 89 onus 1 interactions (15%) and 15 onus 2 interactions (3%). Again, no interactions were rated onus 3.

No new cases were needed to capture the semantic relationships in either experiment. On the other hand, four cases were never assigned in the *clouds* experiment and five were

never assigned in the *small engines* experiment. Section 3.6.2 looks at these cases more closely to determine if they might be superfluous or redundant.

3.6.2 Case System Evaluation

This section revisits the case system evaluation criteria from section 3.3.2. The evaluation is based on the results of the *clouds* and *small engines* experiments and on a survey of the representation of cases among the case markers (Barker 1996). Figure 6 shows the distribution of each case with respect to the prepositional and adverbial markers as well as among the cases assigned in each of the two experiments.

Each case *C* has associated with it four bars in Figure 6. From top to bottom, the bars represent: 1) the percentage of case assignments in the *small engines* experiment accounted for by *C*; 2) the percentage of assignments in the *clouds* experiment accounted for by *C*; 3) the percentage of adverbial marker→case mappings in the case marker dictionary accounted for by *C*; 4) the percentage of prepositional and positional marker→case mappings in the case marker dictionary accounted for by *C*.

For example, Direction accounted for only 0.7% of the cases assigned in the *small engines* experiment and 2.6% of those in the *clouds* experiment, but it is well represented by the markers (20.0% of adverb→case mappings; 8.9% of prepositional or positional mappings). Object is weakly represented by the markers (1.2% of the prepositional or positional mappings and 0.0% of the adverb mappings), yet it accounts for 24.8% of all case assignments in *clouds* and 39.1% of all case assignments in *small engines*.

Generality

In section 3.3.2 I proposed two measures of the generality of a set of cases: 1) the number of times each case is assigned in the analysis of a large body of text; 2) the number of syntactic phenomena (*i.e.*, markers) accounted for by each case.

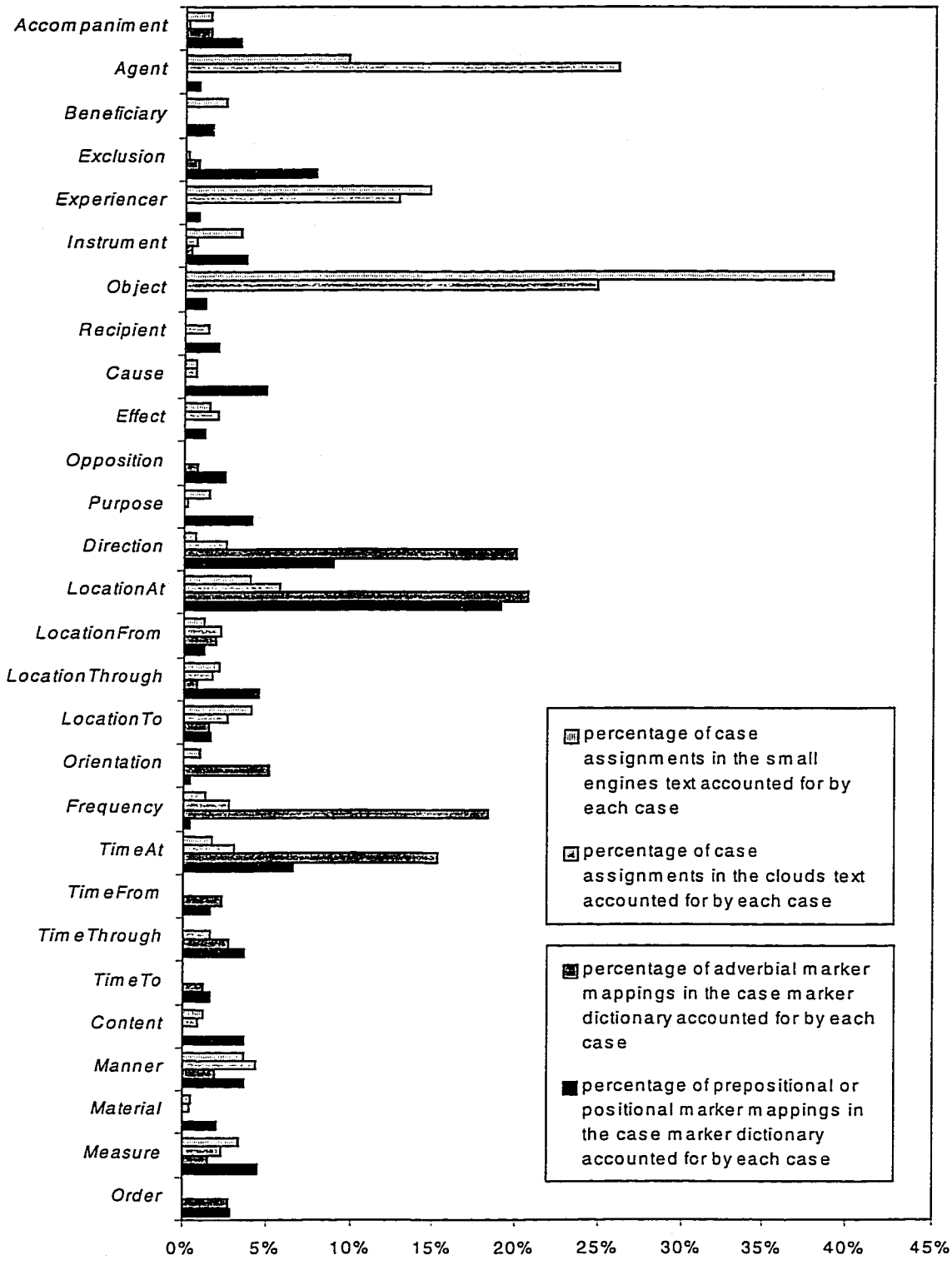


Figure 6: Distribution of the cases in the clouds *and* small engines *experiments, and among the markers*

The *clouds* and *small engines* experiments cannot be considered "a large body of text" for the first measure. Two texts are insufficient for drawing general conclusions about the applicability of each of the cases. For example, below average representation of a particular case in two texts does not necessarily mean that the case is significantly over-specific. Nonetheless, the experiments assigned case patterns to 1023 finite clauses for a total of 2043 individual case assignments in two quite different (though both technical) domains. The data from these assignments are a useful start when investigating the completeness and uniqueness of HAIKU's case set.

The marker→case mapping data charted in Figure 6 can be used to identify the degree to which the markers support the inclusion of each case in the set. Cases whose marker representation is below average are potentially over-specific and may be candidates for combination into a more general case if they are semantically related. Cases whose marker representation is above average are potentially over-general and may be candidates for splitting into more specific cases.

Effect and Purpose are cases with similar definitions and are both weakly represented by the markers (0.6% and 2.0% of all marker→case mappings respectively). These two cases could be merged into a single more general case. They differ in that Effect describes a `+realized` outcome that is `±intended`. Purpose represents a `+intended` outcome that is `±realized`. For applications with no need for such distinctions, a single case might suffice. Effect and Purpose are also weakly represented by the experimental data.

LocationAt and TimeAt are candidates for splitting into more specific cases based on their large number of markers (20.0% and 11.0% of all marker→case mappings respectively). Their representation among the case assignments in the experiments is not nearly so high, with LocationAt accounting for 5.7% of the *clouds* assignments and 4.0% of the *small engines* assignments; TimeAt 3.1% for *clouds* and only 1.8% for *small engines*. One possibility would be to split LocationAt and TimeAt on the `±concrete` feature, though doing so would result in inconsistency with the other locative and temporal cases, whose abstract and concrete interpretations are rolled into individual cases.

Although Agent, Experiencer and Object are weakly represented by the markers, the very high frequency of positional subjects and direct objects in text result in these three cases accounting for a large number of case assignments in practice (*clouds*: 26.0%, 12.7% and 24.8%; *small engines*: 9.7%, 14.7% and 39.1%). The frequency of these cases in texts is not very surprising, since they are most often assigned to required verb arguments.[7]

There are also cases that seem to have general definitions but low marker representation and low occurrence rates in text. For example, the definition of Content allows for both concrete (contents of a container) and abstract (content of communication) interpretations. It could be argued that these are two separate cases. Yet Content accounts for only 1.8% of the marker→case mappings and 0.9% and 1.2% of the *clouds* and *small engines* assignments. Since the two different interpretations seem unlikely to appear together in the same sentence, the Content case might be left as is until more data justify a split.

Completeness

As claimed in section 3.3.2, a practical measure of the completeness of a case set is the number of verb-argument relationships in a text covered by only the cases in the set.

The *clouds* and *small engines* experiments assigned 2043 individual cases in 1023 clauses and did not uncover a verb-argument role not covered by HAIKU's twenty-eight cases. These and other smaller experiments continue to increase the confidence in the completeness of the set.

Uniqueness

The definition of uniqueness from section 3.3.2 requires two things: that the case set have no redundant cases and that it have no superfluous cases.

The presentation of careful case definitions in 3.4.1 is an attempt to avoid redundant cases. The consistent language in these definitions may make them uninteresting, but it

---

[7] Recall that in Cook's Matrix Model (Cook 1989), Object is a required case in all case frames.

helps highlight similarities between cases. Where definitions exposed such similarities, an explicit distinction based on semantic or syntactic grounds was offered with the definitions, as for Agent and Experiencer, Recipient and Beneficiary, Effect and Purpose.

Case marker distribution was also used to avoid redundant and superfluous cases. Cases with identical sets of markers might be redundant; a case whose markers are a strict subset of another case's markers might be superfluous. Marker distribution was the main evidence used in the decision to reject TimeBefore and TimeAfter as superfluous cases.

Example sentences were useful for improving the uniqueness of the case set. In addition to the examples given for each definition in the glossary, an example sentence is stored with every entry in the case marker dictionary (see Appendix II). These example sentences were meant to cover all of the case marking senses of each marker. They provide a justification for the inclusion of a case as marked by a given marker, which in turn justifies the inclusion of the case in the case system.

Redundant Cases

In an experimental setting, redundancy in the case system would be uncovered if, when analyzing a clause, two or more cases were appropriate for a given interpretation of a particular verb-argument relationship. To date, HAIKU's users have not encountered this problem using the existing set of cases. Whenever there has been disagreement among users about which case to assign, it has resulted from ambiguity in the sentence, not redundancy in the case set. Once a single interpretation of an ambiguous sentence has been chosen the choice of cases is unambiguous. For example, (148) has two interpretations depending on whether the noun phrase *various tools* fills the Instrument case or the Material case.

(148) Marty makes sculptures with [various tools ????].

The ambiguity is not due to any redundancy between Instrument and Material. If the intended meaning of (148) is that the sculptures themselves consist of some arrangement of tools, then *various tools* unambiguously fills the Material case. If the intended meaning

is that the tools are used to construct the sculpture (out of wood and stone, for example), then the Instrument case is appropriate.

Note that if two cases can appear in the same clause (as Instrument and Material do in (149)), they are not redundant since a single case may not appear twice in the same clause.

(149) Marty makes sculptures out of [wood and stone $_{MATR}$*] with [various tools* $_{INST}$*].*

Superfluous Cases

Any general purpose, domain independent case system may include cases that are superfluous to the analysis of a particular text. In the *clouds* experiment four cases were not used: Opposition, Order, TimeFrom and TimeTo. The absence of a case in a single text is not evidence that the case is superfluous. In *small engines*, however, *the same four cases were never assigned* (Recipient was also not used). In fact, the temporal cases were in general weakly represented by the data in those experiments.

From its definition, Opposition represents an entity that contrasts with or opposes an event but is insufficient to prevent it from happening. Despite the fact that this case was not assigned in *clouds* or *small engines*, the only other case that might capture the relationship marked by appropriate senses of such markers as *against*, *considering*, *despite*, *in spite of*, *notwithstanding*, *versus*, *nevertheless* and *nonetheless* is Exclusion. (150) shows both Exclusion and Opposition in the same sentence.

(150) They went without [me $_{EXCL}$*] despite [my pleas* $_{OPP}$*].*

Traditional case theory requires that no case appear more than once in a given clause and therefore prevents the merging of Opposition and Exclusion. Exclusion, though, is also weakly represented, with only two assignments in *clouds* and one single assignment (out of more than a thousand) in *small engines*. If Opposition is not theoretically superfluous, it has certainly proven to be practically superfluous. Opposition and Exclusion may well be candidates for merging.

Order represents the relative position of an entity within a structured arrangement of entities involved in an event, as marked by *after*, *ahead of*, *before*, *below*, *beneath*, *beyond*, *underneath*, *first*, *last*, *lastly*, *next*, *primarily*, *second*, *secondly*, etc. The only other case in the set that could capture these relationships is LocationAt. Example (151) contains instances of both LocationAt and Order.

(151) Put the Åkerblads file in [my filing cabinet $_{LAT}$*] after [the A's* $_{ORD}$*].*

The other two cases not used in either experiment were TimeFrom and TimeTo. These are the natural temporal counterparts of the locative LocationFrom and LocationTo cases, both of which appeared frequently in the texts. The TimeFrom and TimeTo cases are also fairly well represented by the case markers (2.0% and 1.4% of marker→case mappings).

Finally, the absence of Recipient in *small engines* suggests that this case might be over-specific, though it did occur fourteen times in *clouds* (1.5%). In case systems that do not contain a Recipient case, the relationship is often covered by some Goal case (Somers 1987). HAIKU's case set does not include a generic Goal case.

3.7 Chapter Summary

This chapter described the construction and evaluation of HAIKU's list of cases. The first step in choosing cases was to reconcile existing lists into a consistent amalgam. A dictionary mapping positional, prepositional and adverbial markers to the cases assists case analysis, but it has also helped expose weaknesses in the case list. The cases were further evaluated on the criteria of generality, completeness and uniqueness using two measures: the number of occurrences of each case in sample texts and the number of marker→case mappings in the case marker dictionary. Evaluation identified cases that might be over-general or over-specific. For the *clouds* and *small engines* experiments, HAIKU's case list was complete, that is, no new cases were required for those texts. The cases were also seen to be unique in that no case's markers are a proper subset of the

markers of some other case and there were no case assignments in the experiments for which more than one case was deemed appropriate.

In this chapter I also reported on performance evaluations of the case analyzer. Experiments showed that as more sentences are analyzed HAIKU learns to make better case pattern suggestions to the user. The average and maximum number of suggestions presented to the user support the claim that case analysis is not overly burdensome. The claim is further supported by the low average onus rating for CA interactions (0.17 average over the *clouds* and *small engines* experiments).

This chapter described the semantic relationships within clauses between a verb and its syntactic arguments, many of which are noun phrases (or contain noun phrases). In the next chapter I go inside the noun phrases to investigate the relationships between a noun and its modifiers.

4 Noun Modifier Relationship Analysis

Noun modifier relationships are semantic relationships between a noun and its modifiers. Syntactic analysis finds noun phrases in a sentence and provides a flat list of premodifiers and postmodifying prepositional phrases and appositives. The noun modifier relationship analyzer first brackets the flat list of premodifiers into modifier-head pairs. Next, it assigns labels to each pair. Labels are also assigned to the relationships with each postmodifying phrase.

This chapter is about HAIKU's third lobe: noun modifier relationship analysis. Since there is little syntactic evidence to guide recognition, the noun modifier relationship analyzer must pay particular attention to user interaction and learning from previous analyses.

4.1 Introduction

This introduction provides background on noun compounds, semantic relations within noun phrases and other systems that recognize these relations in texts. Section 4.2

identifies the parts of a DIPETT parse tree that are relevant to noun modifier relationship analysis (NMRA). In section 4.3 I present HAIKU's semi-automatic premodifier bracketer. I discuss the bracketing problem and kinds of solutions. I also explain why bracketing general noun phrases is not simply an extension of noun-noun-noun bracketing. I then follow with the semi-automatic algorithm, illustrations of the various bracketer features and some difficult examples. The NMR list in section 4.4 is presented through paraphrases and examples. Section 4.5 contains a brief description of the dictionary of mappings from prepositions to NMRs. In section 4.6 I give details of the NMR assignment process: how the similarity of phrases is measured; how the system copes with conflicting previous evidence; the role of the user and user options. Generating taxonomic relationships from NMR structures is also discussed in section 4.6. Results of evaluating both the bracketer and NMR analyzer appear in section 4.7 along with some examples that are not covered by NMRA. The chapter ends with a longer example of bracketing and NMRA applied to several phrases. The example illustrates HAIKU's ability to learn from previous analyses.

4.1.1 Noun Compounds

A head noun along with a premodifying noun is often called a noun compound. Syntactically a noun compound acts as a noun: a modifier or a head may again be a compound. The NMR analyzer deals with the semantics of a particular kind of compound, namely compounds that are *transparent* and *endocentric* (Bauer 1983).

The meaning of a *transparent* compound can be derived from the meaning of its elements. For example, (152) is transparent (a *printer* that uses a *laser*), whereas (153) is *opaque* since there is no obvious direct relationship to *guinea* or to *pig*.

(152) laser printer

(153) guinea pig

An *endocentric* compound is a hyponym of its head. For example, (154) is endocentric because it is a kind of *computer*. (155) is *exocentric* because it does not refer to a kind of *brain*, but rather to a kind of person (whose brain resembles that of a bird).

(154) desktop computer

(155) bird brain

The distinction between endocentric and exocentric is not always obvious, as illustrated by (156), which does not really refer to a kind of *truck*, though it may inherit many of *truck*'s defining features. I will return to examples such as (156) in section 4.7.3.

(156) toy truck

Since the NMR analyzer is intended for semantic analysis of technical text, the restriction to transparent endocentric compounds should not limit the utility of the system. Experiments with the NMR analyzer (such as those described in section 4.7.2) have found no opaque or exocentric compounds in the test texts. For these texts at least, little is lost by not dealing with opaque or exocentric compounds.

4.1.2 Semantic Relations in Noun Phrases

Most of the research on relationships between nouns and modifiers deals with noun compounds, but these relationships also hold between nouns and adjective premodifiers or postmodifying prepositional phrases. Researchers have proposed lists of semantic labels, based on the theory that a compound expresses one of a small number of covert semantic relations between its two elements.

Downing (1977) has noted that the semantic classes of the elements in a compound imply particular semantic interpretations. For example, for head nouns of class *animal*, the preferred interpretation of the modifier is *appearance* or *habitat*. If the head is *synthetic object*, the interpretation of the modifier is *purpose*. Although HAIKU does not have semantic class information for nouns and adjectives, the observation suggests that the relationships likely for a given head noun sense in a compound are restricted and tend to reappear for that sense.

Although it is common to restrict research to noun-noun compounds, Levi (1978) argues that semantics and word formation cause these compounds to constitute a heterogeneous class. Levi removes opaque compounds (such as idioms and compounds that have

become lexicalized) and exocentric compounds from the list. She adds *nominal non-predicating adjectives* to the group as possible modifiers because semantically they function identically to premodifying nouns, as in (157). Levi calls the resulting set of compounds *complex nominals* to distinguish them from noun-noun compounds. She claims that unlike noun-noun compounds, complex nominals form a well defined set.

(157) *[automobile* $_{noun}$*] industry*
[automotive $_{nominal\ non\text{-}predicating\ adjective}$*] industry*

George (1987) questions the claim that Levi's nominal adjectives can never appear in predicative position, offering counterexamples and syntactic tests that show that this *can* happen. Levi admitted the possibility, but said that constructions with non-predicating adjectives in predicative position were the result of more complex transformations, such as ellipsis or head noun deletion. George claims that not all predicative occurrences of non-predicating adjectives can be accounted for by Levi's transformations.

For example, Levi would consider the adjective *civil* in (158) non-predicating, supported by the unacceptable paraphrase in (159). Although the occurrence of *civil* in predicative position in (160) is acceptable, she would claim that it is due to a late deletion in (161) of the second occurrence of *engineer* as head noun.

(158) *civil engineers*

(159) ** the engineers are civil*

(160) *Does your firm employ both civil engineers and electrical engineers?*
No, all of our engineers are civil.

(161) *All of our engineers are civil engineers.*

One of the criticisms of attempts to interpret the semantics of compounds is that the number of possible relationships between the two components is intractable. According to Levi, however, a complex nominal is the result of deleting one of a small number of possible underlying predicates. The interpretation of a complex nominal, then, involves recovering the deleted predicate. The process is multiply ambiguous, but only over the small number of general deletable predicates. Levi lists these *recoverably deletable*

predicates in lexical form, though a typical semantic form is also given. The recoverably deletable predicates appear in Table 20.

Cause	For	Use
Make	About	In
Be	Have	From

Table 20: Recoverably deletable predicates from Levi (1978)

A second kind of complex nominal is the result of predicate nominalization rather than predicate deletion. In a complex nominal formed by predicate deletion, the modifier and head correspond to arguments of the original predicate. In a complex nominal formed by nominalization, the head corresponds to the predicate itself and the modifier corresponds to either the subject or the object of the original predicate. Since the predicate persists as the head noun, complex nominals formed by predicate nominalizations are not restricted to a small number of general predicates like nominals formed by predicate deletion.

Warren (1978) attempts to identify a short list of semantic relations between the constituents of a noun-noun compound. Her *generalized verbs* are meant to represent a predicate that was deleted in the formation of the compound, much like Levi's deleted predicates. For example, a compound expressing a Resemblance relation (such as *pot hole*) is the result of deletion of the *resembles* predicate (in *hole resembles a pot*). Warren also offers a list of prepositional paraphrases for the relationship between elements in the compound. For example, a compound expressing an Origin-Object relation (such as *country boy*) is usually only paraphrased by *from* (as in *boy from the country*).

Warren's semantic relations are not a flat list, but a hierarchy. Each level of the hierarchy gives a different level of detail. The Semantic Class level (below the more general Major Semantic Class and above the more specific Main Group and Subgroup levels) is reproduced as Table 21.

Warren (1984) extends her earlier work to include adjectives as well as modifying nouns. In particular, she extracts from a corpus adjectives with explicit adjectival suffixes such as *-al*, *-an*, *-ar*, *-en*, *-ern*, *-ic*, etc., and identifies which adjectival suffixes are likely to

mark which semantic classes. For example, Place adjectives end only in *-al*, *-an*, *-ar*, *-ern*, *-ese*, *-ic*, *-ish* and *-ly*. Unfortunately, the list of adjectival suffixes under consideration is small and many semantic classes are marked by a majority of them.

Source-Result (constitute)	Copula (be)
Resemblance (be like)	Whole-Part (belong to)
Part-Whole (have in/have on)	Size-Whole (be long, wide, etc./cost/weigh)
Goal-OBJ (lead to)	Place-OBJ (be positioned in/at/on)
Time-OBJ (occur/appear at)	Origin-OBJ (be from)
Causer-Result	Motive Power-Result
Purpose (be for)	Activity-Actor (be concerned with)

Table 21: Noun-noun semantic classes from Warren (1978)

The semantic relations Warren presents for adjectives appear in Table 22. The similarity to the list for noun-noun compounds suggests that many adjectives and premodifying nouns can be handled by the same set of semantic relations.

Source-Result	Origin-OBJ
Result-Source	Time-OBJ
Norm-Adherent	OBJ-Time
Comparant-Compared	Affected Object-Actor
Whole-Part	Causer-Result
Part-Whole	Result-Causer
Place-OBJ	Purpose

Table 22: Adjective-noun semantic classes from Warren (1984)

4.1.3 Recognizing Semantic Relations

Programs that uncover the relationships in modifier-noun compounds often base their analysis on the semantics of the individual words, which assumes the existence of some dictionary of semantic information.

Leonard's system (1984) assigns semantic labels automatically to noun-noun compounds. The semantic relationships appear in Table 23. To arrive at an interpretation, the system proceeds through nine ordered heuristics that test semantic features of the nouns in the compound. The program depends on the semantic features encoded manually in Leonard's dictionary. These features include taxonomic information, syntactic behaviour,

information about relationships between nouns and verbs, relative size of a concept in a meronymic hierarchy, etc. For example, if the modifying noun has a semantic feature *material* and the head has a semantic feature *can be composed of material*, then the semantic relationship between the two is likely Material. Experiments reveal a 76% accuracy in assigning relationships to previously unseen compounds consisting of known words (*i.e.*, words in Leonard's dictionary).

Sentence	Locative Sentence	Locative	Annex
Equative	Material	Additive	Reduplicative

Table 23: Noun-noun semantic relationships from Leonard (1984)

Finin's UNCLE system (1986) deals with the semantic interpretation of nominal compounds, which include both noun-noun pairs and nominal adjective-noun pairs. UNCLE produces multiple possible interpretations of a compound and assigns an appropriateness score to each. The interpretations are based on precoded semantic class information and domain-dependent frames describing the roles that can be associated with certain nouns. For example, for the compound *Chicago flight*, UNCLE gives three possible interpretations: *Chicago* fills either a Source role, a Destination role or a Stopover role. These interpretations are based on knowledge of both *Chicago* and *flight*: *Chicago* is known to be a city, and *flight* is a frame with Source, Destination and Stopover slots whose fillers are restricted to cities.

Ter Stal (1996) describes the interpretation of nominal compounds in the Plinius project. Interpretation unifies compounds with concept structures extracted from a hand-coded lexicon. For each word the lexicon contains part of speech information, "other syntactic features" (including subcategorization information, agreement features, etc.), an Underspecified Logical Form for the word, and the concept to which the word refers within an ontology. The restricted domain and corpus size (400 abstracts on mechanical properties of ceramic materials) limit the amount of hand-coding necessary, but building the ontology and lexicon still requires considerable labour and expertise. The availability of the hand-coded information allows ter Stal to skirt several linguistic problems. For example, he investigates various methods for bracketing three-word compounds but

ultimately treats multi-word compounds as flat lists, since the information available in the lexicon and ontology allow direct identification of complex concepts.

In an attempt to avoid the hand-coding required by other systems, Vanderwende (1993) automatically extracts semantic features of nouns from online dictionaries. SENS (the System for Evaluating Noun Sequences) chooses the most likely semantic relation between two nouns in sequence by considering the semantic properties of each noun. The features extracted from dictionary definitions and example sentences appear in Table 24.

Abstract	Has-Subject	Material
Caused-By	Human	Means
Event	Is-For	Object-Of
Food	Location-Noun	Subject-Of
Group-Noun	Location-Of	Time
Has-Object	Made-Of	

Table 24: Semantic features extracted from dictionaries in Vanderwende (1993)

Combinations of the semantic features imply particular semantic interpretations. For example, if the head noun has the *IS-FOR* feature and the modifier matches one of the values of that feature, then the semantic relationship between head and modifier is almost certainly a What for? relationship (Purpose). SENS also attempts matching on hypernyms of the nouns (if available from the dictionary extraction) if matching on the nouns themselves fails. The semantic relations used in SENS appear in Table 25.

Who/what? (Subject of a deverbal head)	How? (Instrument)
Whom/what? (Object of a deverbal head)	What for? (Purpose)
Where? (Locative)	What kind of? (Equi/General)
When? (Time)	Made of what? (Material)
Whose? (Possessive)	What does it cause? (Causes)
	What causes it? (Caused-by)

Table 25: Semantic relations between nouns in Vanderwende (1993)

Instead of proposing his own arbitrary list, Lauer (1995a) does away with abstract semantic relations altogether. Explicit paraphrases, clauses or prepositional phrases, usually help illustrate relationships in noun-noun compounds. Lauer wants these paraphrases to be the semantic relations. He does not claim that paraphrasing as semantic

analysis is easier or more difficult than assigning abstract relationship labels, and admits that the two problems are analogous. He does say that paraphrases are more concrete. Lauer's eight "concrete semantic relations" are *of*, *for*, *in*, *at*, *on*, *from*, *with*, *about*, borrowed largely from Warren's identification (1978) of prepositional paraphrases for her semantic relations. Semantic analysis assigns probabilities to each of the different possible paraphrases of a compound. These probabilities are based on the probability distribution of the relations in which the *modifier* is likely to participate and those in which the *head* is likely to participate. The frequency of prepositional phrases in a large corpus is used to estimate the probabilities.

Fabre (1996) performs semantic interpretation of noun-noun compounds using semantic characteristics of the nouns taken from WordNet (Miller 1990). Compounds that contain a deverbal noun refer to either the accomplishment of the act corresponding to the verbal root of the noun, or one of its thematic roles. If the compound does not contain a deverbal noun, a covert predicate (verb) is assumed and both nouns fill a thematic role of that covert predicate. Roles are Agent, Theme, Instrument, Locative, etc. The covert predicates are either specific to a particular compound or general enough to apply to several nouns. For example, the compound *pipeline* has the specific covert predicate Transport. More general predicates are Characterize, Constitute, Contain and Measure.

4.2 Input Structures

The input to noun modifier relationship analysis in HAIKU is a noun phrase structure (called an *entity* structure in DIPETT). The structure has a number of arguments:

```
entity( intensifier,
        determinatives,
        attributes,
        head_noun(noun( head noun,
                        noun_modifiers( premodifiers, postmodifiers )))
        number,
        noun phrase postmodifiers)
```

A noun phrase may also be a proper noun or a bare possessive. Proper nouns are not analyzed for internal structure. Bare possessives have no modifiers and therefore do not participate in NMRs. Details of the entity structure appear in Barker (1997).

Prior to parsing a noun phrase, DIPETT also performs morphological analysis, so that all words arrive at NMRA in a root form (plural to singular, gerunds and participles to root verb, comparative and superlative adjectives to positive form, etc.).

The various elements of the noun phrase identify *where* the semantic relationships exist—*i.e.*, which syntactic elements are involved in a noun modifier relationship. Despite the variety of noun phrase elements that the noun phrase structure can represent, few of the elements are involved in noun modifier relationships. Only attributes, head noun premodifiers and noun phrase postmodifiers contain elements that enter into NMRs. Unfortunately, the structure of these elements offers little indication of what those relationships are.

4.2.1 Attributes and Head Noun Premodifiers

DIPETT organizes modifying elements preceding the head noun into two flat lists: *attributes* and *noun premodifiers.* Originally, these two groups corresponded exclusively to adjectives and premodifying nouns respectively, based on the assumption that all attributive position adjectives precede all premodifying nouns. This assumption is valid for *attributive-only* (non-predicating) *adjectives*, as supported by the acceptable phrase (162) and the questionable phrase (163):

(162) a [fast $_{adj}$] [desktop $_{noun}$] [computer $_{head\ noun}$]

(163) ? a [desktop $_{noun}$] [fast $_{adj}$] [computer $_{head\ noun}$]

For non-predicating adjectives, the assumption does not hold: both (164) and (165) are acceptable. To reflect this fact, the DIPETT grammar has been changed to accept adjectives among the noun premodifiers.

(164) a [personal $_{adj}$] [desktop $_{noun}$] [computer $_{head\ noun}$]

(165) a [desktop $_{noun}$] [personal $_{adj}$] [computer $_{head\ noun}$]

Most noun compound research makes more subtle distinctions on premodifiers. Recall that Levi (1978) allows nouns and nominal, non-predicating adjectives as premodifiers in complex nominals, and so does Finin (1986). Lauer (1995a) points out that trying to classify adjectives as predicating and non-predicating causes computational difficulties. He restricts his compounds to sequences of "nouns acting as a single noun", as do Downing (1977) and Leonard (1984).

An attribute or premodifier recognized by DIPETT may be any string accepted as an adjective or noun. There is no distinction between predicating and non-predicating adjectives, nor is that information readily available from an outside source (such as an online dictionary or wordlist). The NMRs and NMR analyzer may have to account for any adjective or noun premodifier, including words that act as adjectives or nouns (such as participles, gerunds, etc.)

Before assigning NMRs, the system concatenates the attributes, premodifiers and head noun into a single flat list. It must then bracket this list into local modifier-head pairs using the semi-automatic bracketing techniques from section 4.3. Each modifier-head pair receives an NMR assignment.

4.2.2 Noun Phrase Postmodifiers

DIPETT recognizes four main types of noun phrase postmodifier: the *relative clause*, *appositives*, *restricted comparison* and *prepositional phrases*.

Relative Clauses

A *relative clause* postmodifying a noun phrase can be any finite clause recognized by DIPETT. Since it is a finite clause, it is case analyzed; the noun phrase is assigned a case within the relative clause. The case structure of the relative clause provides the link between the noun phrase and the postmodifying clause, making assignment of an NMR to the postmodifier unnecessary. Noun phrases within the relative clause are analyzed separately by NMRA.

Appositives

Appositives come in many different forms and serve many different roles as postmodifiers of noun phrases. Quirk *et al.* (1985: 17.66-68) identify different features and degrees of apposition. Regardless of the type of apposition, though, all appositives are normally considered coreferent. For this reason, the same NMR (Equative) will always be assigned to the relationship between a head noun and a postmodifying appositive. Noun modifier relationships within appositives will receive NMR analysis separately.

Comparison

DIPETT allows a single restricted type of comparison to appear as a noun phrase postmodifier, illustrated by example (166).

(166) She owns [a faster machine $_{np}$] [than mine $_{compar}$].

Since the noun modifier relationships are not intended to cover general comparative forms, there is no NMR that is appropriate for the comparative in (166). Experiments with the NMR analyzer to date have encountered no comparatives as noun phrase postmodifiers.

Prepositional Phrases

A noun phrase may have any number of postmodifying prepositional phrases. Multiple PPs may be either explicitly conjoined, as in (167), or implicitly conjoined, as in (168). Implicitly conjoined PPs are assumed to be conjunctive.

(167) Use the paper [[in the drawer $_{pp/LOC}$] or [on the counter $_{pp/LOC}$] $_{conj\_pp}$].

(168) the printer [[in the lab $_{pp/LOC}$] [on the desk $_{pp/LOC}$] $_{conj\_pp}$]

(169) the printer [[with the broken tray $_{pp/CONT}$] [in the lab $_{pp/LOC}$] $_{conj\_pp}$]

Conjunctive conjoined prepositional phrases may express the same NMR or different NMRs, as in (168), where both PPs express the Location NMR, or (169), where the *with* PP expresses Content and the *in* PP expresses Location. It appears that disjunctive

conjoined prepositional phrases are unlikely to express different NMRs, and this guess is supported by the questionable (170). Nevertheless, there is no guarantee that such examples will *not* occur in texts. Accordingly, each PP in an explicit disjunction undergoes individual NMR assignment.

(170) ? Use the paper [[in the paper tray $_{pp/LOC}$*] or [with the fancy border* $_{pp/CONT}$*]].*

4.3 Noun Modifier Bracketing

An NMR is a binary relationship between a noun and a modifier. Often, the head of a noun phrase is preceded by a sequence of modifiers. In this sequence modifiers may modify the head directly, or they may modify other modifiers. Before assigning NMRs, HAIKU must bracket the sequence into nested modifier-head pairs. Example (172) shows the bracketing for noun phrase (171).

(171) dynamic high impedance microphone

(172) (dynamic ((high impedance) microphone))

The bracketing problem has been investigated by Liberman & Sproat (1992), Pustejovsky *et al.* (1993), Resnik (1993) and Lauer (1995b) among others. Systems that bracket premodifiers usually deal with the problem of bracketing three nouns: two premodifiers and a head (X-Y-Z). Bracketers typically compare occurrences of X-Y with occurrences of either Y-Z or X-Z (depending on the model—see section 4.3.2). Since these pairs may occur infrequently in a text, it would be useful to generalize (X-Y-Z) and look for occurrences of the generalization. For example, Lauer (1995b) generalizes the nouns X, Y and Z to the Roget's Thesaurus categories that contain them: R_X, R_Y and R_Z. Instead of looking for other occurrences of X-Y and X-Z in the text, Lauer's bracketer looks for occurrences of U-V and U-W such that $R_U = R_X$, $R_V = R_Y$ and $R_W = R_Z$. The technique limits generalization to nouns that occur in Roget.

4.3.1 Subphrases and Reduced Subbracketings

The problem of generalization is compounded in HAIKU since it deals with sequences of more than three words and allows adjectives as premodifiers. And since the fundamental tenet of HAIKU is to avoid precoding semantic information, a semantic generalization on the words is not readily available. We need to increase the number of hits when we search for previously analyzed noun phrases to help analyze a new noun phrase.

Consider phrase (173) and a reasonable bracketing for it in (174).

(173) dynamic high impedance vocal microphone

(174) (dynamic ((high impedance) (vocal microphone)))

Each non-atomic element of each bracketed pair can be considered a subphrase of the original phrase. Given the bracketing in (174) the subphrases for phrase (173) are phrase (173) itself as well as the subphrases in (175). If the compounds are endocentric, the subphrases will be well-formed.

(175) high impedance vocal microphone
high impedance
vocal microphone

Each subphrase consists of one or more modifiers and a head local to the subphrase. Local heads in (173) are *microphone* (the head of three subphrases) and *impedance*. The subphrases can be generalized by reducing each modifier and modificand to their local heads. The result is a set of reduced subbracketings of the original phrase. The reduced subbracketings of bracketed phrase (174) appear in (176).

(176) (dynamic microphone)
(impedance microphone)
(high impedance)
(vocal microphone)

The reduced subbracketings together are a structural generalization of the original noun phrase. Instead of simply memorizing complete noun phrases and their analyses, the system stores the subbracketings. This allows it to analyze different noun phrases that have only subbracketings in common with previous noun phrases.

4.3.2 Bracketing Models

A decision whether a given sequence X-Y-Z is left-branching, as in (177), or right-branching, as in (178), may be reached in two ways.

(177) ((laser printer) manual)

(178) (desktop (laser printer))

Liberman & Sproat (1992), Pustejovsky *et al.* (1993) and Resnik (1993) all compare the number of isolated occurrences of X-Y in a corpus with the number of occurrences of Y-Z. Lauer (1995b) calls this the *adjacency* model and offers a different model, the *dependency* model, which compares the number of occurrences of X-Y to the number of occurrences of X-Z (rather than Y-Z). In Lauer's bracketer, the dependency model outperforms the adjacency model. Experiments by ter Stal (1996) using both models tend to confirm Lauer's results, yet ter Stal also notes that the results are not appreciably better than always guessing left-branching.

HAIKU's bracketer uses the dependency model, comparing frequencies of X-Y to X-Z when it brackets the sequence X-Y-Z, ignoring previous occurrences of Y-Z. Consider phrase (179). Both left- and right-bracketings are possible.

(179) small business loan

(180) (small (business loan))

(181) ((small business) loan)

Previous occurrences of *small loan* would be evidence for right-bracketing. Occurrences *small business* would be evidence for left-bracketing. Occurrences of *business loan*, on the other hand, could be evidence for *either* bracketing: (180) restricts the *loan*; (181) restricts the *business*. But both contain *(business loan)* as a reduced subbracketing—both refer to some kind of *business loan*. Therefore, occurrences of Y-Z do not count as evidence in favour of either bracketing.

4.3.3 A Bracketing Algorithm

Bracketing sequences of three elements requires a single branching decision. That branching decision leads directly to a bracketing of the compound. Bracketing sequences of more than three elements requires multiple branching decisions. Each decision does not map directly to a bracketing. Rather, decisions are interdependent.

Example (182) requires a single branching decision to arrive at the left-bracketing shown. Examples (183) and (184) both add *top* as a single modifier in front of the sequence in (182). In both the new examples *ring pin location* still branches left. But the ultimate bracketing of (183) and (184) does not directly contain the bracketing from (182). So even once *ring pin location* is known to be left-branching, it cannot be bracketed until a subsequent branching decision involving *top* has been made. A general bracketing algorithm therefore can not be a trivial iterative (or recursive) application of an algorithm for bracketing sequences of three elements.

(182) ((ring pin) location)

(183) (((top ring) pin) location)

(184) ((top (ring pin)) location)

HAIKU's algorithm for noun premodifier bracketing handles modifier sequences of any length by dealing with a sliding/expanding window of three elements at a time.[1] Depending on the results of individual branching decisions, the system may have to delay bracketing locally within the window until later branching decisions have been made.

1 Start with the rightmost three elements, X-Y-Z.

...	V	W	X	Y	Z

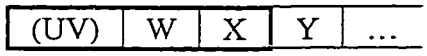
2a If X-Y-Z is confidently right-branching (see section 4.3.4), bracket it X-(YZ) and restart the algorithm with the rightmost three elements W-X-(YZ).

...	V	W	X	(YZ)

[1] An element is a word or a bracketed pair of elements.

2b If X-Y-Z is confidently left-branching (see sections 4.3.4 and 4.3.6), move the window one element to the left and repeat the algorithm with W-X-Y. Note that X-Y-Z being confidently left-branching does not necessarily mean that X and Y can be immediately bracketed together as (XY)-Z; the bracketing ((WX)Y)-Z also has X-Y-Z left-branching.

3 When the leftmost element in the whole sequence appears in the window, a left-branching sequence U-V-W can be left-bracketed (UV)-W; restart with the three-element window expanded to include the next element to the right in the sequence.



When bracketing is complete, each reduced subbracketing is stored to assist future bracketing. The entire bracketed phrase is also stored, and can be looked up if the same noun phrase occurs again.

4.3.4 Confidence in Branching Decisions

The bracketing algorithm needs to know whether subsequences X-Y-Z in the modifier sequence are confidently left-branching or right-branching.

The sequence X-Y-Z is confidently right-branching when Y is an adjective, since adjectives are almost always the modifier in a modifier-head pair. There are exceptions, where the adjective appears in the head position of modifier-head pairs. Section 4.3.8 addresses bracketing exceptions such as (185) and (186).

*(185) ((machine **readable**) dictionary)*

*(186) ((ill **educated**) man)*

For any other sequence of three elements X-Y-Z, the bracketer reduces X, Y and Z to their local heads X_h, Y_h and Z_h. The sequence X-Y-Z is considered confidently right-branching if the frequency of previous occurrences of the reduced subbracketing (X_hZ_h) is greater than the frequency of previous occurrences of the reduced subbracketing (X_hY_h)

times a threshold value. If (X_hY_h) has occurred more frequently than (X_hZ_h) times the threshold, X-Y-Z is confidently left-branching.

In the absence of such frequency data, the algorithm consults the user. A fully automatic solution could just assume left-branching, based on the results of Lauer & Dras (1994) and ter Stal (1996) that left-branching is roughly twice as common as right-branching. These results, however, were based on observations of noun-noun-noun sequences. For longer compounds and compounds with adjectives, the ratio of left-branching to right-branching compounds might differ (see section 4.7.1).

4.3.5 User Interaction

When the system cannot find sufficient evidence in favour of right-branching or left-branching, it asks the user to supply the decision. Such decisions may be difficult and unintuitive for users. There are several ways to lessen the burden.

- ask only yes-no questions about right-branching—don't ask the user to supply bracketing information directly.

good: in the context of *copper soup pot*,
does *copper soup* make sense?

bad: does *copper soup pot* bracket left or right?

- phrase questions in the context of three individual words by using subphrase reductions.

good: in the context of *steel soup pot*,
does *steel soup* make sense?

bad: in the context of *steel tomato soup pot cover*,
does *steel tomato soup pot* make sense?

- ask the user only about the acceptability of X-Y; do not ask the user to compare X-Y and X-Z—it is possible for both X-Y and X-Z to be unacceptable if X-Y-Z is in the middle of a modifier sequence, which would make the user's decision much more difficult.

good: in the context of *steel tomato soup*,
does *steel tomato* make sense?

bad: in the context of *steel tomato soup*,
which makes more sense: *steel tomato* or *steel soup*?

Using these principles, a 'yes' answer to any question will provide confident left-branching; a 'no' answer will mean confident right-branching.

4.3.6 When Is "Left-Branching" "Not-Right-Branching"?

I have been using the terms *right-branching* and *left-branching* for subsequences of a premodifier sequence being bracketed. Although right-branching was appropriate in the context, left-branching was less so. Consider phrase (187).

(187) French onion soup bowl

The bracketing algorithm starts with the rightmost sequence of three elements *onion soup bowl*, which is *not* right-branching. But the subphrase *onion soup bowl* in the context of (187) is not immediately left-branching, since *onion soup* is the head of the right-branching compound *French onion soup*, as shown graphically in Figure 7. Nonetheless, preferring simplicity over exactness, I refer to not-right-branching subsequences as left-branching.

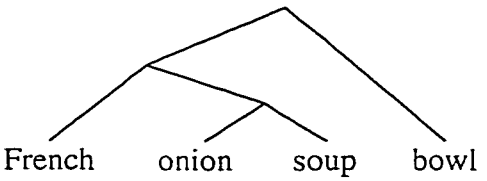


Figure 7: Branching for (187)

4.3.7 A Walk through Bracketing

I will now walk through the application of the bracketing algorithm to a longish example (190) with some previous bracketings (188) and (189) available. Figure 8 shows the actual user interaction for this example using the implemented bracketer.

(188) (soup bowl)

(189) (wooden (pot handle))

(190) wooden French onion soup bowl handle

Start with the rightmost three elements, *soup-bowl-handle.*

wooden	French	onion	soup	bowl	handle

soup-bowl-handle is confidently left-branching, since *(soup bowl)* has occurred and *(soup handle)* has not. Move the window one element to the left and restart the algorithm.

wooden	French	onion	soup	bowl	handle

Neither *(onion soup)* nor *(onion bowl)* have occurred previously and *soup* is not an adjective, so there is no confidence in right-branching or left-branching. Ask the user if *onion soup* makes sense in the context of *onion soup bowl.* The user answers 'yes', providing confidence in left-branching. Move the window one element to the left and restart the algorithm.

wooden	French	onion	soup	bowl	handle

```
String (190)       wooden French onion soup bowl handle

HAIKU: Noun Modifier Relationship Analysis of current input ...

> For the phrase 'onion soup bowl'
  is that 'onion soup' [Y/n]? Y

> For the phrase 'french onion soup'
  is that 'french onion' [Y/n]? n

> For the phrase 'wooden soup bowl'
  is that 'wooden soup' [Y/n]? n

    ____________________________
    |                          |
  wooden                       ____________________
                               |                  |
                               |                handle
                          _____________
                          |           |
                          |          bowl
                     _____________
                     |           |
                  french         _____________
                                 |           |
                               onion        soup

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort  Y
```

Figure 8: User interaction for phrase (190)

Neither *(French onion)* nor *(French soup)* have occurred previously. Ask the user if *French onion* makes sense in the context of *French onion soup*. The user answers 'no', so the sequence is confidently right-branching. Bracket *(onion soup)* and expand the window to include the next element to the left.

wooden	French	(onion soup)	bowl	handle

French is an adjective, so *wooden-French-(onion soup)* is confidently right-branching. Bracket *(French (onion soup))*. Since there are no more elements to the left of *wooden*, expand the window to include the next element to the right.

wooden	(French (onion soup))	bowl	handle

Neither *(wooden bowl)* nor *(wooden soup)*, which is the reduction of *wooden-(French (onion soup))*, have occurred previously. *(French (onion soup))* is obviously not an adjective, so there is no confidence in either right-branching or left-branching. Ask the user if *wooden soup* makes sense in the context of *wooden soup bowl*, which is the reduction of *wooden-(French (onion soup))-bowl*. The user answers 'no', providing confidence in right-branching. Bracket *wooden-(French (onion soup))-bowl* as *wooden-((French (onion soup)) bowl)*. Since there are no more elements to the left of *wooden*, expand the window to include the next element the right.

wooden	((French (onion soup)) bowl)	handle

(wooden bowl), which is the reduction of *wooden-((French (onion soup)) bowl)*, has not occurred previously; *(wooden handle)* has occurred previously as a reduction of *(wooden (pot handle))*. So *wooden-((French (onion soup)) bowl)-handle* is confidently right-branching. Bracket *(((French (onion soup)) bowl) handle)*.

wooden	(((French (onion soup)) bowl) handle)

Since there are only two remaining elements in the sequence, bracketing is trivial:

(wooden (((French (onion soup)) bowl) handle))

Finally, the system stores all of the reduced subbracketings (191) for future processing. It also stores the complete bracketing (192): if the phrase is ever encountered in its entirety, the bracketer can simply look up the complete bracketing.

(191) *(onion soup)*
(French soup)
(soup bowl)
(bowl handle)
(wooden handle)

(192) *(wooden (((French (onion soup)) bowl) handle))*

4.3.8 When All Else Fails

The bracketer may make incorrect branching decisions based on previous analyses. There may be head-position adjectives in the premodifier sequence (see section 4.3.4). There may be bracketings in the text that conflict, or the user may simply have incorrectly answered one of the branching questions.

Redoing the Bracketing Interaction

The semi-automatic bracketer always submits the final bracketing to the user for approval. The user may accept the bracketing, or choose from several other options. The simplest option is to repeat the bracketing interaction using the same evidence as the first time through the process. This option is useful for redoing the bracketing when questions may have been answered incorrectly.

Ignoring Bracketing History

Certain previous modifier-head phrases may "fool" the bracketer. In this situation, no matter how the user answers questions, the final bracketing is incorrect. The user must turn off the bracketing history and redo the interaction as though there were no previous examples. For example, assume phrases (193) has already been bracketed.

(193) *(metric (socket wrench))*

(194) *metric system wrench*

Implementation Note

History *on/off* is a parameter that can be set in user profiles for HAIKU. If history is *on* for the current session, the user can turn it *off* for an interaction, but it is automatically turned back *on* for subsequent noun phrases. If the user has history set *off* in the profile, it can be turned *on* for individual interactions, but is automatically turned *off* for subsequent noun phrases.

When bracketing (194) the system compares the number of previous occurrences of *metric system* (none) with the number of occurrences of *metric wrench* (one). It incorrectly concludes that (194) is confidently right-branching and does not ask the user to make the branching decision. The user can redo the interaction with history off to arrive at the correct bracketing, as shown in Figure 9.

```
String (193)       metric socket wrench

HAIKU: Noun Modifier Relationship Analysis of current input ...

> For the phrase 'metric socket wrench'
  is that 'metric socket' [Y/n]? n

     ________________
     |              |
  metric       _____________
               |           |
            socket      wrench

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort Y

...

String (194)       metric system wrench

HAIKU: Noun Modifier Relationship Analysis of current input ...

     ________________
     |              |
  metric       _____________
               |           |
            system      wrench

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort h

> For the phrase 'metric system wrench'
  is that 'metric system' [Y/n]? Y

        _________________
        |               |
  _____________       wrench
  |           |
  |           |
  metric    system

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory on, (m)anual, (a)bort Y
```

Figure 9: An incorrect bracketing repaired by turning history off

Evaluation of the bracketer on complete English texts (see section 4.7.1) has shown that situations where previous analyses mislead the bracketer are rare. In the *small engines* experiment history had to be turned off only twice in 118 complete interactions.

Bracketing by Hand

While turning off history may allow the system to recover from some bracketing errors, there are phrases that foil the bracketing algorithm altogether. One example is an adjective appearing in a local head position in the modifier of a noun, as in (195).

(195) machine readable dictionary

The bracketer assumes that an adjective Y in the sequence X-Y-Z signals right-branching. Even with history turned off, the heuristic persists. To bracket such phrases the user must bypass the semi-automatic bracketer altogether and use the manual bracketer.[2]

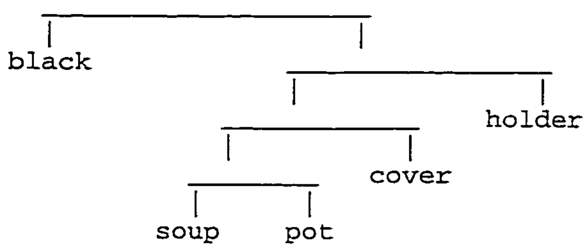
The manual bracketer allows the user to enter bracketing information directly in the form of bracket characters and numbers corresponding to words in the phrase. The system has two parts: a *list lexicalyzer* and a *list parser*. The lexicalyzer ensures that the bracketing list typed by the user consists of only brackets, commas, whitespace and digits. It also ensures that only and all the appropriate digits appear in the correct (consecutive increasing) order. The parser builds a bracketed version of the modifier list parallel to the input list while ensuring that the input list is a correct binary bracket list. The parser also treats whitespace, commas and the outermost brackets as optional. Figure 10 shows a sample user interaction with the manual bracketer for phrase (196).

(196) black soup pot cover holder

---

[2] Example (195) is also difficult for NMR assignment. I will return to it in section 4.7.3).

```
Manual Modifier Bracketing Editor:
> Please enter the bracketing for the current NP using the word
  numbers in order.  The outermost brackets are optional.
     E.g.:  [ 1 [2 3] ] 4

1   black
2   soup
3   pot
4   cover
5   holder

|: 1 [[[2 3] 4] 5]
```




```
> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort Y
```

Figure 10: Manual bracketing interaction for (196)

Once again, experiments have shown that situations where the manual bracketer is required occur rarely. In the *small engines* experiment, the manual bracketer was used three times:

(197) ((fully trained) mechanic)

(198) ((badly worn) cylinder)

(199) ((badly worn) (valve seat))

In each phrase, the intensifier (*fully*, *badly*) was misparsed as an adjective. A leading adverb should be parsed as an *intensifier* (in the entity structure—see section 4.2). Intensifiers should be concatenated with the first adjective and treated as a single modifier in NMRA (since there are no NMRs to handle the relationship between intensifier and adjective).

4.4 The Noun Modifier Relationships

Table 26 lists HAIKU's noun modifier relationships. The list was constructed after carefully considering similar lists found in literature on the semantics of noun compounds (section 4.1).

Agent (AGT)	Instrument (INST)	Property (PROP)
Beneficiary (BENF)	Located (LED)	Purpose (PURP)
Cause (CAUS)	Location (LOC)	Result (RESU)
Container (CTN)	Material (MATR)	Stative (STAT)
Content (CONT)	Object (OBJ)	Source (SRC)
Destination (DEST)	Possessor (POSS)	Time (TIME)
Equative (EQUA)	Product (PROD)	Topic (TOP)

Table 26: The noun modifier relationships (with abbreviations)

Some comments on the list are in order. First, some researchers name the semantic relationships with two role labels. For example, a PartOf relationship might be named Part-Whole, with the modifier labeled Part and the head labeled Whole. This could be justified. If a modifier-head pair hides some deleted predicate (see Levi 1978), then there may not be a single relationship between the modifier and the noun, but rather one semantic relationship between the modifier and the deleted predicate and one between the head and the deleted predicate. Nonetheless, this complex set of relationships can be captured by a single abstract label for the relationship, even if all the details are not explicit in the label. For example, a PartOf relationship label denotes a Part and implies a Whole.

Second, many lists of semantic relationships mark the direction of the relationship by including separate labels for each direction. For example, a Part-Whole label (modifier: Part, head: Whole) might have a corresponding Whole-Part label (modifier: Whole, head: Part) in some list. While this is justified for some relationships (for example, Located and Location from Table 26), not all relationships have such natural inverses. Others' lists (for example, Warren 1978; 1984) confirm the asymmetry. Rather than introduce new labels exhaustively for the inverses, I will be conservative and add them individually if experiments reveal the need (see also section 5.3.2).

Finally, although the literature that has inspired the NMRs usually rules out certain kinds of modifiers—some ignore predicative adjectives, some ignore adjectives altogether, some avoid nominalizations, some appositional (karmadharaya) compounds—the TANKA context allows no such luxury. DIPETT passes all different kinds of modifiers to NMRA, and distinctions such as the non-predicating versus predicating feature of adjectives are not available anywhere. For these reasons, the list of NMRs contains some relationships not always found elsewhere. Property covers predicative adjectives; Equative is used for appositional compounds and Stative for other possibly copula-derived compounds. The case-like NMRs (Agent, Beneficiary, Cause, etc.) allow for nominalizations.

4.4.1 NMR Glossary

The glossary (Table 27) contains paraphrases and example compounds for the NMRs. Following the tradition in the study of noun compound semantics, the paraphrases act as definitions and can be used to check the acceptability of a semantic interpretation of a compound. The paraphrases serve as definitions in this section and as help during user interactions (as illustrated in section 4.6.5).

In the analyzer, paraphrases with adjectives may feel awkward; they could be improved by replacing adjectives with their WordNet pertainyms, giving, for example, (202) as the paraphrase of (200) instead of the infelicitous (201).

(200) charitable donation

(201) charitable benefits from charitable donation

(202) charity benefits from charitable donation

NMR	*Paraphrase*	*Examples*
Agent	*compound* is performed by *modifier*	*student protest, band concert, military assault*
Beneficiary	*modifier* benefits from *compound*	*student price, charitable donation*
Cause	*modifier* causes *compound*	*exam anxiety, overdue fine*
Container	*modifier* contains *compound*	*printer tray, flood water, film music, story idea*
Content	*modifier* is contained in *compound*	*paper tray, eviction notice, oil pan*
Destination	*modifier* is the destination of *compound*	*game bus, exit route, entrance stairs*
Equative	*modifier* is also *head*	*composer arranger, hinge joint, player coach*
Instrumental	*modifier* is used in *compound*	*electron microscope, diesel engine, laser printer*
Located	*modifier* is located at *compound*	*building site, home town, solar system*
Location	*modifier* is the location of *compound*	*lab printer, internal combustion, desert storm*
Material	*compound* is made of *modifier*	*carbon deposit, gingerbread man, water vapour*
Object	*modifier* is acted on by *compound*	*engine repair, horse doctor*
Possessor	*modifier* has *compound*	*national debt, student loan, company car*
Product	*modifier* is a product of *compound*	*automobile factory, light bulb, colour printer*
Property	*compound* is *modifier*	*blue car, big house, fast computer*
Purpose	*compound* is meant for *modifier*	*concert hall, soup pot, grinding abrasive*
Result	*modifier* is a result of *compound*	*storm cloud, cold virus, death penalty*
Source	*modifier* is the source of *compound*	*foreign capital, chest pain, north wind*
Stative	*compound* is in a state of *modifier*	*live wire, sleeping dog, charged battery*
Time	*modifier* is the time of *compound*	*winter semester, late supper, morning class*
Topic	*compound* is concerned with *modifier*	*computer expert, safety standard, horror novel*

Table 27: NMRs with paraphrases and examples

4.5 The NMR Marker Dictionary

Unlike clause level relationships and cases, noun modifier relationships are not always marked by a lexical or syntactic element. NMRs between heads and postmodifiers are marked lexically by a preposition or syntactically as appositives. Between premodifiers and heads, no such NMR markers exist.[3] The preposition attaching a postmodifier to its head can be used to reduce the number of potential NMRs, based on the assumption that each preposition marks a subset of the NMRs.

The NMR marker dictionary contains an entry for each of 46 atomic and phrasal prepositions. These prepositions were taken from several online word lists. The lists were first merged and then edited by hand to remove noise (including archaic words, uncommon foreign language prepositions, errors, etc.). For each final entry the various English dictionary senses of the preposition were mapped to NMRs by hand. This step constitutes hand-coding of semantic information, but since the list of prepositions is small, the mapping was not a difficult knowledge engineering task. I enlisted a secondary school student to build the dictionary; he completed the task in just twenty intensive hours.

The total number of mappings in the NMR marker dictionary is 126, meaning that each preposition marks on average 2.7 NMRs. 15 prepositions are mapped to a single NMR, 14 are mapped to only two. The preposition *of* is the least discriminating, mapping to 12 NMRs. A sample of the entries in the NMR marker dictionary appears in Appendix III.

The marker dictionary was constructed carefully, but it is possible that there are missing entries. Identifying new mappings in examples from texts would be simple: they consist of those prepositional phrases for which the dictionary was consulted but for which the user supplied the correct NMR.

---

[3] I do not consider simple adjacency as a syntactic NMR marker, though that argument could be made. A *bracketing* is a marking in the sense that it indicates which elements participate in semantic relationships. The bracketing itself, however, does not help disambiguate in NMR assignment.

4.6 Assigning NMRs

The process of assigning NMRs is as follows. For the modifier and head under consideration, find the most similar previously analyzed instances. Create a list (or lists) of the NMRs that were assigned to these previous instances. Suggest the best NMRs from the list(s) to the user. The user may accept the system's suggestion or supply a different NMR.

4.6.1 Reduced Modifiers and Heads

After bracketing, each non-atomic element of a bracketed pair is considered a subphrase of the original phrase, as described in section 4.3.1. Consider again the bracketed phrase (172), reproduced here with its subphrases (203), (204) and (205).

(172) (dynamic ((high impedance) microphone))

(203) high impedance

(204) (high impedance) microphone

(205) dynamic (high impedance microphone)

Each subphrase consists of a modifier (possibly compound, as in (204)) and a head (possibly compound, as in (205)). The NMR analyzer assigns an NMR to the modifier-head pair that makes up each subphrase.

Once an NMR has been assigned, the system must store the assignment to help automate future processing. Instead of memorizing complete noun phrases (or even complete subphrases), the system reduces compound modifiers and compound heads to their own local heads and stores these reduced pairs with their assigned NMR. For example, (206) and (207) have the reduced pair *(dynamic microphone)* in common. So if (206) has already been analyzed, its analysis can be used to assist in the analysis of (207).

(206) (dynamic ((high impedance) microphone))

(207) ((dynamic (cardioid microphone)) diaphragm)

Although the system stores *reduced* pairs for future processing, it assigns NMR labels to *complete* modifier-head sequences. Consider example (208).

(208) (small (gasoline engine))

There are two NMRs for (208), since there are two modifier-head pairs:

(209) gasoline engine

(210) small (gasoline engine)

The NMR for (209) could be Instrument, since *gasoline* is used by *gasoline engine*. Note that *gasoline* is an instrument of *gasoline engine*, not *engine* in general. The NMR for (210) is Property: *small* is a property of *small gasoline engine*, but not generally a property of *engine* or even of *gasoline engine*. When an argument name is built from more than one word, the individual words are concatenated with an underscore. The result of NMR analysis for (208) would be:

gasoline is used in *gasoline_engine*
small_gasoline_engine is *small*

4.6.2 Modifier-Head-Marker Triples

Section 4.2 identified three kinds of construction that require NMR assignments: the modifier-head pairs from the bracketed premodifier sequence; postmodifying prepositional phrases; appositives.

These three kinds of input can be generalized to a single form—a triple consisting of modifier, head and marker (M, H, Mk). For premodifiers, Mk is the symbol `nil`, since no lexical item links the premodifier to the head. For postmodifying prepositional phrases Mk is the preposition. For appositives, Mk is the symbol `appos`. The (M, H, Mk) triples for examples (211), (212) and (213) appear in Table 28.

(211) monitor cable plug

(212) large piece of chocolate cake

(213) my brother, a friend to all young people

To assign an NMR to a triple (M, H, Mk), the system attempts to find previous triples whose distance to the current triple is minimal. The NMRs assigned to previous similar triples comprise lists of candidate NMRs. The analyzer then finds what it considers the best NMR from these lists of candidates and presents it to the user for approval. Appositives are automatically assigned Equative.

	modifier	*head*	*marker*
(211)	monitor	cable	nil
	monitor_cable	plug	nil
(212)	chocolate	cake	nil
	large	piece	nil
	chocolate_cake	large_piece	of
(213)	young	people	nil
	young_people	friend	to
	friend	brother	appos

Table 28: (M, H, Mk) triples for (211), (212) and (213)

4.6.3 Distance between Triples

The distance between two triples is a measure of the degree to which their modifiers, heads and markers match. Table 29 gives the eight different values for distance used in NMRA. An underscore in a previous triple means that the modifier, head or marker need not be the same as the corresponding modifier, head or marker in the current triple.

dist	*current triple*	*previous triple*	*example*	
0	(M, H, Mk)	(M, H, Mk)	*wall beside a garden*	*wall beside a garden*
1	(M, H, <prep>)	(M, H, nil)	*wall beside a garden*	*garden wall*
2	(M, H, Mk)	(M, H, _)	*wall beside a garden*	*wall around a garden*
3	(M, H, Mk)	(M, _, Mk) or (_, H, Mk)	*pile of garbage*	*pile of sweaters*
4	(M, H, <prep>)	(_, _, <prep>)	*pile of garbage*	*house of bricks*
5	(M, H, <prep>)	(_, _, _)	*ice in the cup*	`nmrm(in, [ctn,...,time])`
6	(M, H, Mk)	(M, _, _) or (_, H, _)	*wall beside a garden*	*garden fence*
7	(M, H, Mk)	(_, _, _)	*wall beside a garden*	*pile of garbage*

Table 29: Measures of distance between triples

The analyzer looks for previous triples at the lower distances before attempting to find triples at higher distances. For example, it will try to find identical triples (distance 0) before trying to find triples whose markers do not match (distance 1, 2, ...).

Several things about the distance measures require explanation. First, a preposition is assumed to be more similar to a `nil` marker (distance 1) than to a different preposition. The `nil` marker could be considered a variable, which is not known to be different from the marker in an overtly marked pair. For example, (214) could be paraphrased using the marker *about* from (215) or the marker *on* from (216). (215) and (216), on the other hand, are known to have different markers.

(214) computer encyclopedia

(215) encyclopedia about computers

(216) encyclopedia on computer

Next, no evidence suggests that triples with matching modifiers are more similar or less similar than triples with matching heads. Consequently, previous triples with matching modifiers but different heads are at the same distance as previous triples with matching heads but different modifiers (distances 3 and 6).

Triples with matching prepositional markers (distance 4) *are* considered more similar than triples with matching modifiers or heads only. A preposition is an overt indicator of the relationship between modifier and head (see Quirk 1985: chapter 9) so a correlation is more likely between the preposition and the NMR than between a given modifier or head and the NMR.

If there are no matching triples at a distance of 4 or less and the marker is a preposition, HAIKU consults the NMR marker dictionary for candidate NMRs (represented by `nmrm(...)` in the distance 5 row of Table 29).

4.6.4 The Best NMRs

The candidate NMRs are all NMRs previously assigned to (M, H, Mk) triples at a minimum distance from the triple under analysis. If the minimum distance is 3 or 6, there

may be two candidate lists: $\mathbf{L_M}$ contains the NMRs previously assigned to triples with matching M, $\mathbf{L_H}$—with matching H. HAIKU attempts to choose a set **R** of NMRs to suggest to the user as the best candidates for the current triple.

If there is one list **L** of candidate NMRs, **R** contains the NMR (or NMRs) that occur most frequently in **L**. For two lists $\mathbf{L_M}$ and $\mathbf{L_H}$, **R** could be found in several ways.

Suppose the modifier-head pair in example (217) has never been seen before, but that *front* has appeared as the modifier in three other pairs (218) and *panel* has appeared as the head in nine other pairs (219).

(217) front panel

(218) [front cover $_{LOC}$], [front line $_{LOC}$], [front plate $_{LOC}$]

(219) [computer panel $_{CTN}$], [control panel $_{PURP}$], [control panel $_{PURP}$], [glass panel $_{MATR}$], [plastic panel $_{MATR}$], [plastic panel $_{MATR}$], [side panel $_{LOC}$], [steel panel $_{MATR}$], [wood panel $_{MATR}$]

Absolute Frequency

One possibility is to take **R** to contain the most frequent NMR(s) in $\mathbf{L_M} \cup \mathbf{L_H}$. Table 30 shows the absolute frequencies of each of the NMRs in $\mathbf{L_M} \cup \mathbf{L_H}$. Material has the highest absolute frequency in (218) and (219). Using absolute frequency, NMRA would suggest the following analysis of (217) to the user:

front_panel is made of *front*

NMR	*absolute frequency*
Container	1
Location	4
Material	**5**
Purpose	2

Table 30: Best NMR for (217) using the absolute frequency method

Relative Frequency

The absolute frequency method has a bias towards NMRs in the larger of the two lists. Alternatively, HAIKU could prefer NMRs with the highest relative frequency in their lists.

If there is less variety in the NMRs in $\mathbf{L_M}$ than in $\mathbf{L_H}$, M might be a more consistent indicator of NMR than H (or vice versa). Table 31 shows the relative frequency of each NMR i in each list calculated as $freq(i \in \mathbf{L_M})/|\mathbf{L_M}|$ and $freq(i \in \mathbf{L_H})/|\mathbf{L_H}|$.

NMR	*relative frequency in* $\mathbf{L_M}$	*relative frequency in* $\mathbf{L_H}$
Container	0.0	0.11
Location	**1.0**	0.11
Material	0.0	0.56
Purpose	0.0	0.22

Table 31: Best NMR for (217) using the relative frequency method

R contains the NMR(s) with the highest relative frequency in either $\mathbf{L_M}$ or $\mathbf{L_H}$. There is a potential bias, however, for smaller lists (a single NMR in a list always has the highest relative frequency). Using absolute frequency, NMRA would suggest the following analysis of (217) to the user:

front is the location of *front_panel*

Weighted Relative Frequency

The relative frequency method might be going too far in ignoring absolute frequency altogether. A third possibility would be to combine absolute and relative frequencies. Each NMR i is assigned a score s_i calculated as:

$$s_i = \frac{freq(i \in \mathbf{L_M})^2}{|\mathbf{L_M}|} + \frac{freq(i \in \mathbf{L_H})^2}{|\mathbf{L_H}|}$$

Table 32 shows the weighted relative frequency score for each of the NMRs in $\mathbf{L_M} \cup \mathbf{L_H}$.

NMR	*weighted score*
Container	0.11
Location	**3.11**
Material	2.78
Purpose	0.44

Table 32: Best NMR for (217) using weighted relative frequency scores

R would contain the NMR(s) with the highest score. This method was used in the *small engines* experiment. Experimental results appear in section 4.7.2.

4.6.5 User Interaction

For each NMR assignment the NMR analyzer asks the user's approval showing the paraphrases from section 4.4.1. The user has several options. Most often over the course of a session, the user accepts the suggestion. Alternatively, he may supply an NMR directly, ask for a list of NMR paraphrases using the current modifier and head, or even create a new NMR. Figure 11 illustrates use of the `list` feature for example (220).

(220) steel casing

```
String (220)      steel casing

HAIKU: Noun Modifier Relationship Analysis of current input ...

Match type 7: (_, _, _)

For the phrase 'steel casing'
There is a relationship between
   (steel) and (steel_casing).

> Please enter a valid NMR label ('a' to abort): list

  Agent (agt): steel_casing is performed by steel
  Beneficiary (benf): steel benefits from steel_casing
  Cause (caus): steel causes steel_casing
  Container (ctn): steel contains steel_casing
  Content (cont): steel is contained in steel_casing
  Destination (dest): steel is the destination of steel_casing
  Equative (equa): steel is also casing
  Instrument (inst): steel is used in steel_casing
  Located (led): steel is located at steel_casing
  Location (loc): steel is the location of steel_casing
  Material (matr): steel_casing is made of steel
  Object (obj): steel is acted on by steel_casing
  Possessor (poss): steel has steel_casing
  Product (prod): steel is a product of steel_casing
  Property (prop): steel_casing is steel
  Purpose (purp): steel_casing is meant for steel
  Result (resu): steel is a result of steel_casing
  Source (src): steel is the source of steel_casing
  State (stat): steel_casing is in a state of steel
  Time (time): steel is the time of steel_casing
  Topic (top): steel_casing is concerned with steel

> Please enter a valid NMR label ('a' to abort): matr

> For the phrase 'steel casing'
  Do you accept the assignment:
  Material (matr): steel_casing is made of steel

   [n/a/<nmr>/Y] Y
```

Figure 11: The `list` feature for example (220)

HAIKU also allows the user to add a new NMR. Figure 12 shows an interaction using the `create` feature for example (221). The `create` feature has a particularly unfriendly interface, requiring the user to enter such things as the NMR argument ordering for paraphrases. This feature is considered the last resort for particularly tricky phrases.[4]

(221) water faucet

```
String (221)      water faucet

HAIKU: Noun Modifier Relationship Analysis of current input ...

Match type 7: (_, _, _)

For the phrase 'water faucet'
There is a relationship between
   (water) and (water_faucet).

> Please enter a valid NMR label ('a' to abort): create

You're not supposed to create new NMRs.

> Please enter the NMR Label (3-4 lower case letters): isrc
> Please enter the NMR Name: Inv_Source
> Please enter the NMR argument order (mh, cm, etc.): cm
> Please enter the NMR paraphrase: is the source of

I'm about to add your NMR to the NMR list:
   t_nmr_names(isrc,'Inv_Source',cm,' is the source of ')
This is your last chance to back down.
> Do you accept the new NMR [y/N]? Y

New NMR saved (but I'm not happy about it).

> For the phrase 'water faucet'
  Do you accept the assignment:
  Inv_Source (isrc): water_faucet is the source of water

   [n/a/<nmr>/Y] Y
```

Figure 12: The `create` feature for example (221)

4.6.6 Classifying Function of Premodifiers

Since NMR analysis deals with endocentric compounds, it is possible to recover a taxonomic relationship from modifier-head-marker triples with a `nil` marker (*i.e.*, those that come from compounds). The reduced pairs for example (222) appear in (223).

[4] See section 5.3 for a discussion of the possibility of adding NMRs such as Inv_Source.

(222) (colour (laser printer))

(223) (laser printer)
(colour printer)

These pairs result in the additional HAIKU output:

laser_printer isa *printer*
colour_laser_printer isa *laser_printer*

The `isa` relationships are automatically generated for all of the NMRs with one somewhat arbitrary exception. Property relationships do not result in `isa` structures, because properties do not introduce hyponyms: a *small house* is not generally considered a subclass of *house*, for example. A more general solution would be to associate a feature with each NMR giving its "`isa` pattern". This simple extension is explored in section 4.7.3 and again in section 5.3.

4.7 Evaluation

In this section I give results of two experimental evaluations of NMR analysis. The *sparc* experiment applied the NMR analyzer (including the bracketer) to the first 500 non-trivial noun phrases in the *sparc* text. In this context a non-trivial noun phrase has at least one premodifier (adjective or noun) or postmodifying prepositional phrase. Bracketing the 500 noun phrases in the test produced 637 bracketed pairs. These bracketed pairs along with 129 postmodifiers were assigned 766 NMR labels. The second experiment is the *small engines* experiment. Bracketing resulted in 733 premodifier-head pairs. Along with 153 postmodifiers, a total of 886 NMRs were assigned.

Implementation Note

The NMR analyzer stores three kinds of structures: `nmsDict` structures (containing reduced pairs that have occurred in bracketings and the number of times they have occurred); `nmsDictWhole` structures (complete flat lists of premodifier-head sequences and their corresponding bracketed forms); `nmrDict` structures (M-H-Mk triples with reduced M and H along with the NMRs assigned to them). These are the structures used to assist processing new phrases. At the end of a session the user has the option of saving the structures in a file. At the beginning of a session the user may load any number of such files of structures to seed the new session.

4.7.1 Bracketer Evaluation

System Performance

In both experiments, most of the modifier-head pairs occurred in noun phrases with a single premodifier and head. These simple compounds required no bracketing decisions. In the *sparc* experiment, the 637 pairs required 194 bracketing decisions. The system made 122 (63%) of these decisions correctly, with the rest made by the user. Of the 72 user decisions, 47 (65%) were required during the first half of the experiment with only 25 in the second half. The running totals of user and system bracketing decisions appear in Figure 13.

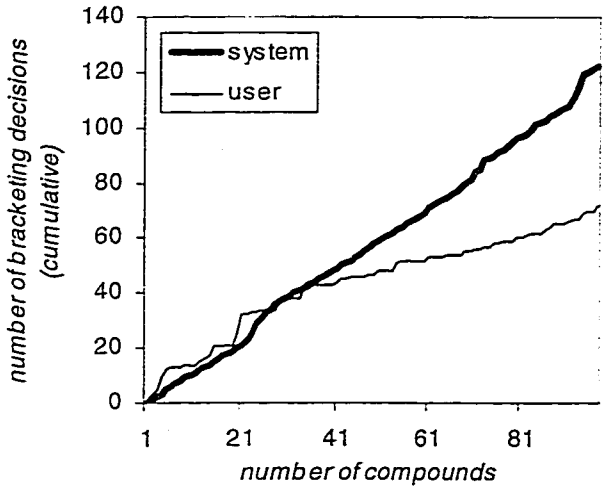


Figure 13: Bracketing decisions in the sparc *experiment*

The *small engines* experiment required 164 bracketing decisions. The system made 101 (62%) decisions correctly, with the rest made by the user. Due to the consistency of the terminology in the *small engines* text the cumulative number of decisions taken automatically by the bracketer was always greater than the number required from the user. The running totals for user and system bracketing decisions appear in Figure 14.

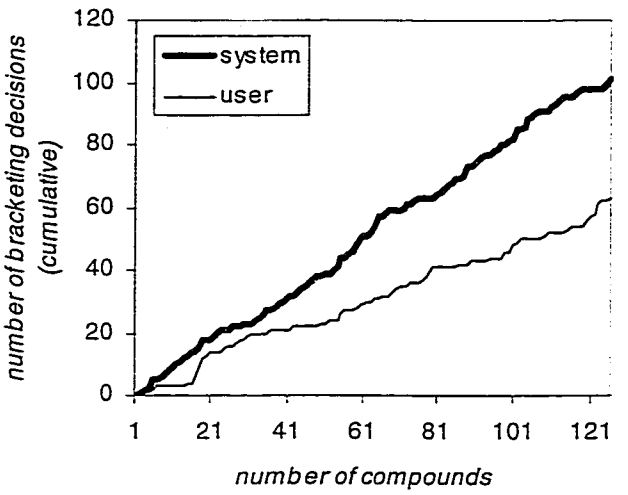


Figure 14: Bracketing decisions in the small engines *experiment*

The Effect of the Threshold

As explained in section 4.3.4, the bracketer determines that a given sequence X-Y-Z is confidently right-branching if the subbracketing (X_hZ_h) has previously occurred more frequently than the subbracketing (X_hY_h) by a factor of N, where N is a threshold that can be set by the user. In the absence of sufficient evidence, the system asks the user questions to help acquire right-branching information.

If the value of the threshold is set high, the number of previous occurrences of (X_hZ_h) must greatly outweigh the number of occurrences of (X_hY_h) for the system to assume right-branching. So high values of the threshold cause the system to be more conservative. Low values of the threshold (close to 1.0), make the system more aggressive: HAIKU requires less evidence to commit to a branching decision automatically.

The bracketer was run on the *sparc* phrases twelve times with different threshold values. As expected, the number of system decisions, both correct and incorrect, was highest for low threshold values (see Figure 15). For higher threshold values, the number of incorrect

system decisions decreased, but so did the number of correct decisions, the extra decisions being given to the user.

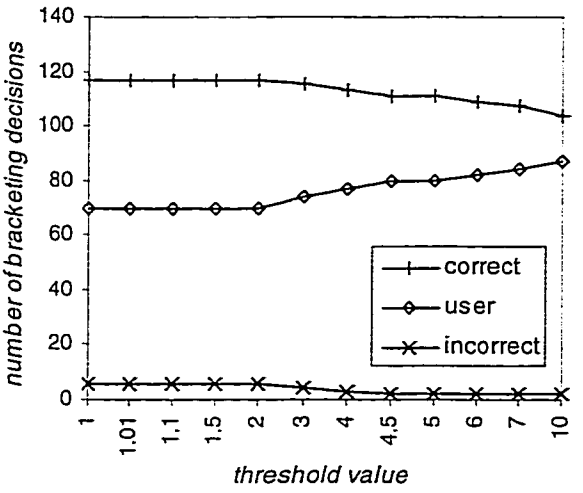


Figure 15: The effect of different threshold values on branching decisions for the sparc *experiment*

For the *small engines* experiment, changing the threshold had no effect. This result suggests that for any sequence X-Y-Z in the *small engines* text, if (X_hZ_h) appears as a reduced pair, (X_hY_h) does not, but it is not a universal observation.

Branching Frequencies

Ter Stal (1996) confirms earlier results from Resnik (1993) and Lauer & Dras (1994) that between 60% and 70% of noun-noun-noun compounds in text are left-branching. Section 4.3.4 suggested that the bracketer could guess left-branching when there is no confidence in right-branching. Such a guess would be justified assuming the predominance of left-branching compounds. Results from the *small engines* experiment confirm the bias for left-branching.

Implementation Note

For testing purposes, the NMR analyzer allows the optional storage of bracketing structures and NMR structures indexed by noun phrase number. Bracketing and NMR assignment proceed as usual, using information from preceding analyses only. But whenever the system *would have* consulted the user, it consults the indexed structures for the current phrase instead. After a text has been analyzed once, it can be reanalyzed fully automatically to test different threshold values, different "best NMR" formulae, etc.

	left-branching	*right-branching*
noun-noun-noun	47 (96%)	2 (4%)
adjective-noun-noun	31 (84%)	6 (16%)
total	78 (91%)	8 (9%)

Table 33: Branching frequencies for the small engines *text*

For the *sparc* experiment, however, the data in Table 34 show that guessing left-branching would have produced poor results.

	left-branching	*right-branching*
noun-noun-noun	41 (55%)	33 (45%)
adjective-noun-noun	11 (26%)	31 (74%)
total	52 (45%)	64 (55%)

Table 34: Branching frequencies for the sparc *text*

The predominance of left-branching compounds is apparently not universal. If the system were modified to guess left in the absence of other evidence, there are texts (like the *sparc* text) for which the bracketer would perform poorly.

4.7.2 NMRA Evaluation

System Performance

The *sparc* experiment assigned NMRs to 766 modifier-head pairs. The system's assignment is considered *correct* when the user accepts its single suggestion *or* chooses one from among its multiple suggestions. According to this definition, the system assigned 555 NMRs (72%) correctly, with the user supplying 211 labels. For 532 of the system's correct assignments (96%) it offered a single suggestion. For the 23 multiple choice suggestions, there were an average of 3.3 suggested NMRs.

Of the 500 noun phrases, 311 were unique after morphological analysis. Assuming a similar distribution, one could extrapolate that a system learning only by accumulating analyses of complete noun phrases could perform automatically no more than 38% of the time. Even using reduced pairs, only 433 of the 766 assignments (57%) were distance 0 matches. The partial matching techniques (section 4.6.3) increased the number of correct analyses by 15%.

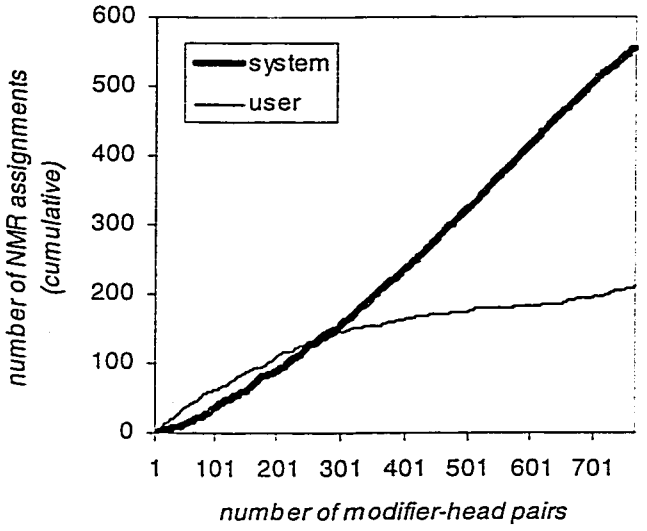


Figure 16: NMR assignments in the sparc *experiment*

Figure 16 shows the cumulative number of NMR assignments supplied by the user versus those determined correctly by the system. After about 300 assignments, the system was able to make the majority of assignments automatically. The curves in the figure show that the system learns to make better suggestions as more phrases are analyzed.

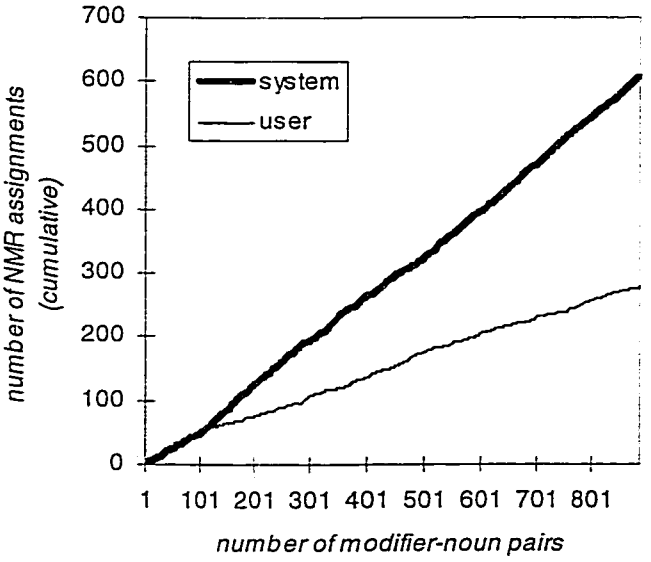


Figure 17: NMR assignments in the small engines *experiment*

In the *small engines* experiment (Figure 17), the system assigned NMRs to 886 modifier-noun pairs. 608 of the 886 NMRs (69%) were assigned correctly by the system. For 586 of these assignments (97.5%) the system offered a single suggestion. It had multiple suggestions (on average 3.3 again) only 22 times. There were 384 distance 0 matches (43%), meaning that partial matching increased the number of correct assignments by another 26%.

Both the *sparc* and *small engines* experiments used a version of the NMR analyzer with the *weighted relative frequency* method of choosing best NMRs from two candidate lists (see section 4.6.4). A second experiment on the *sparc* text applied five different methods to distance 3 and distance 6 triples (those that might generate two lists of candidate NMRs: L_M and L_H). In addition to the *absolute frequency* and *weighted relative frequency* methods from section 4.6.4, the implemented system has a *basic union* method, a *modifier preference* method and a *head preference* method.

Whereas the *absolute frequency* method takes the most frequent NMRs in $L_M \cup L_H$, the *basic union* method takes *all* of the NMRs in $L_M \cup L_H$ (without duplicates) and presents them to the. The *modifier preference* method presents the most frequent NMRs in L_M only, ignoring L_H. *Head preference* uses L_H and ignores L_M. The five methods were applied to 324 distance 3 or distance 6 triples in the *sparc* text. Results appear in Table 35.

	accept	*choose (average)*	*supply*
basic union	101	125 (2.8)	98
absolute frequency	168	22 (2.3)	134
weighted relative	174	16 (2.2)	134
modifier preference	182	12 (2.3)	130
head preference	143	21 (2.3)	160

Table 35: Applying different "best NMR" methods to sparc

As usual, *accept* represents the number of times a given method identified a single NMR as the best NMR and it was the correct one. *Choice* is the number of times the method identified 2 or more NMRs and the correct NMR was among them; *(average)* is the

average number of best NMRs identified. The *supply* column shows the number of times the correct NMR was not among those chosen by the given method.

The results are inconclusive. The *absolute frequency* and *weighted relative* methods perform about equally well. *Modifier preference* performs better than *head preference*, though there is no theoretical reason to expect this result. It is difficult to compare the *basic union* method to the others. It resulted in far fewer user-supplied NMRs, but the number of multiple choice interactions was six times higher than for the other methods.

Coverage

As with the other HAIKU modules, NMRA is affected by the quality of DIPETT parse trees, though to a lesser degree. During the *small engines* experiment we noted that several noun phrases in sentences did not receive NMR analysis because of errors in the parses for those sentences (Barker *et al.* 1998). It would not be feasible to compare directly the number of NMR assignments made by HAIKU to the number of relationships in the text—even in a text of only a few hundred sentences like *small engines*.

To measure coverage, I sampled 100 modifier-noun pairs at random from the *small engines* text and found that 87 of them appeared in HAIKU's output. At the 95% confidence level, the system extracted between 79.0% and 92.2% of the modifier-noun relationships in the text.

User Burden

To measure the burden that NMR analysis places on the user, we assigned an onus rating to each interaction during the *small engines* experiment. As explained in section 1.4.4, the onus is an integer from 0 to 3, with 0 assigned to the simplest interactions and 3 to the most arduous. The average user onus rating was 0.1 for NMR interactions in the *small engines* experiment. 808 of the 886 NMR assignments received an onus rating of 0; 71 had a rating of 1; 7 received a rating of 2. No interactions were rated onus level 3.

4.7.3 Some Difficulties

Questionably Endocentric Compounds

Section 4.1.1 restricted NMRA to transparent endocentric compounds. In that section I also pointed out that the line between endocentric and exocentric is sometimes blurred. For example, it is unclear whether compounds such as those in (224) are endocentric or exocentric.

(224) toy truck, teddy bear, stuffed animal, gingerbread man, fake gun, chocolate bunny

If the compounds are taken to be endocentric they pose no problem: assign them Equative, Equative, Property, Material, Property and Material respectively. The NMR paraphrases are shown in (225).

(225) *toy* is also *truck*
teddy is also *bear*
stuffed_animal is *stuffed*
gingerbread_man is made of *gingerbread*
fake_gun is *fake*
chocolate_bunny is made of *chocolate*

This interpretation has precedence. Leonard (1984) assigns her Material label to (226). Lauer (1995a) claims that (227) and (228) are adequately interpreted as equative and material[5] relationships, even if metaphorically.

(226) stone lion

(227) barrel chest

(228) steel father

Kamp (1975) takes the opposite position. He argues that compounds such as those in (225) are purely exocentric—that a *fake gun* is by definition an entity that is never a *gun*. Franks (1995) distinguishes two kinds of *privative* adjective: *negators* (such as *fake*) and

---

[5] Lauer does not actually use relationship labels such as Equative and Material. He does refer to (227) as a copula compound that can be paraphrased *chest that is a barrel.* His paraphrase for (228) is *father of steel.*

equivocators (such as *apparent*). Negators contradict defining features of the head noun; equivocators merely undermine them.

Warren (1978) introduces a special relation Resemblance, with the paraphrase *be like*. There is no reason why such a relationship could not be added to HAIKU's list of NMRs. (229) shows the possible paraphrases for the compounds in (224).

(229) *toy_truck* is like a *truck*
teddy_bear is like a *bear*
stuffed_animal is like an *animal*
gingerbread_man is like a *man*
fake_gun is like a *gun*
chocolate_bunny is like a *bunny*

The problem with assigning Resemblance to all of these examples is that the interpretations under the existing NMRs are valid: a *chocolate bunny* really is made of *chocolate*. Where the examples fail is not in the assignment of NMRs, but in the generation of `isa` structures based on the assumption that they are purely endocentric. One solution is to allow the user to defeat generating the `isa` structure for each NMR assignment. A second solution would be to invent a new generalization structure that is weaker than `isa`. It would again be up to the user to override `isa` generation in favour of an `islikea` structure (for example).

I have decided against making `isa` generation interactive. In the *small engines* experiment, HAIKU automatically generated 431 `isa` structures, all of them appropriate. Asking the user to approve `isa` generation would have required another 431 strokes of the Enter key. For less technical texts (with more examples of the (224) variety) the extra user interaction might be justified.

Implementation Note

The `isa` structure generator already checks the NMR before generating `isa` structures. Property NMRs do not result in `isa` structures. New NMRs (such as Resemblance) could trivially be added to the list of non `isa`-generating NMRs. Only slightly less trivial would be the addition of an `isatype` field to each NMR: {`isa`; `islikea`; `none`}.

Postpositive Adjectives

The manual bracketer (section 4.3.8) allows for the left-bracketing of phrases such as (230).

(230) machine readable dictionary

Once it has been bracketed, however, example (230) poses a problem for NMR assignment. NMRA assumes that its compounds represent entities, not properties. No NMR adequately expresses the relationship in *machine readable*, and certainly none of the paraphrases work. And even if the user did assign an NMR, the `isa` structure would be inappropriate.

Fortunately, the pair noun-adjective (or adjective-adjective) as a modifier within a compound is often hyphenated to avoid ambiguity (see Fowler 1984). Example (231)[6] is ambiguous precisely because of the two possible bracketings (232). Hyphenating disambiguates.

(231) ill educated man

(232) ((ill educated) man)
(ill (educated man))

If the pair is hyphenated, DIPETT will treat it as a single modifier, allowing NMRA to proceed normally. If there is no hyphen, the user will have to bracket manually and then abort NMR analysis for the postpositive adjective, because no NMR is appropriate. NMR assignment for the rest of the elements in the phrase is still possible, as shown in Figure 18 for example (230).

---

[6] from Fowler (1984: 256).

```
String (230)       machine readable dictionary

HAIKU: Noun Modifier Relationship Analysis of current input ...

      ____________________
     |                    |
  machine          _______________
                  |               |
              readable     dictionary

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort m

Manual Modifier Bracketing Editor:
> Please enter the bracketing for the current NP using the word
  numbers in order.  The outermost brackets are optional.
     E.g.:  [ 1 [2 3] ] 4

1   machine
2   readable
3   dictionary

|: [1 2] 3

          ____________________
         |                    |
     ____________          dictionary
    |            |
  machine     readable

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort Y

Match type 7: (_, _, _)

For the phrase 'machine readable'
There is a relationship between
   (machine) and (machine_readable).

> Please enter a valid NMR label ('a' to abort): a

NMR Analysis for this pair aborted by user.

Match type 7: (_, _, _)

For the phrase 'machine_readable dictionary'
There is a relationship between
   (machine_readable) and (machine_readable_dictionary).

> Please enter a valid NMR label ('a' to abort): stat

> For the phrase 'machine_readable dictionary'
  Do you accept the assignment:
  State (stat): machine_readable_dictionary is in a state of
                machine_readable

   [n/a/<nmr>/Y] Y
```

```
NOUN MODIFIER RELATIONSHIPS
  State (stat): machine_readable_dictionary is in a state of
                machine_readable

  machine_readable_dictionary isa dictionary
```

Figure 18: Bracketing and NMRA for a postpositive adjective

Adjectives in Paraphrases

Section 4.4.1 suggested that WordNet could be used to improve awkward paraphrases involving adjectives. The awkward paraphrase (234) of example (233) could be fixed by using the WordNet pertainym for *solar*, which is *Sun*.

(233) solar system

(234) solar is located at solar system

(235) Sun is located at solar system

Although pertainyms would make NMRA interaction smoother, I have chosen *not* to implement this feature. The overhead of the WordNet in Prolog API would be justified by using WordNet in the many other parts of HAIKU that could benefit from it. That system would be a very different HAIKU than the one described here. Identifying all of those parts (including pertainyms in paraphrases) and implementing a full HAIKU-with-WordNet would be a good future project (see section 5.5.2).

4.8 An Example

In this section I show the HAIKU NMRA interaction for four noun phrases, starting with no previous bracketings or analyses:

(236) small gasoline engine

(237) the repair of diesel engines

(238) diesel engine repair shop

(239) an auto repair centre

Of course, it is unlikely that these four noun phrases would occur isolated and in sequence in a text. Nonetheless, noun phrases of this kind, scattered throughout the *small engines* text, illustrate well the use of previous phrases to help bracketing and NMR assignment.

Since (236) is the first noun phrase, no previous bracketed pairs or NMR assignments can help analysis. The user has to make bracketing decisions and supply NMR labels directly. Results of the interaction are two NMR assignments and only one `isa` relationship (recall that `isa` relationships are suppressed for Property).

```
String (236)      small gasoline engine

HAIKU: Noun Modifier Relationship Analysis of current input ...

> For the phrase 'small gasoline engine'
  is that 'small gasoline' [Y/n]? n

    _________________
   |                 |
  small          ___________
                 |           |
            gasoline     engine

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort Y

Match type 7: (_, _, _)

For the phrase 'gasoline engine'
There is a relationship between
   (gasoline) and (gasoline_engine).

> Please enter a valid NMR label ('a' to abort): inst

> For the phrase 'gasoline engine'
  Do you accept the assignment:
  Instrument (inst): gasoline is used in gasoline_engine

   [n/a/<nmr>/Y] Y
```

```
Match type 7: (_, _, _)

For the phrase 'small gasoline_engine'
There is a relationship between
   (small) and (small_gasoline_engine).

> Please enter a valid NMR label ('a' to abort): prop

> For the phrase 'small gasoline_engine'
  Do you accept the assignment:
  Property (prop): small_gasoline_engine is small

   [n/a/<nmr>/Y] Y

NOUN MODIFIER RELATIONSHIPS
  Property (prop): small_gasoline_engine is small
  Instrument (inst): gasoline is used in gasoline_engine

  gasoline_engine isa engine
```

Figure 19: NMRA interaction for (236)

For phrase (237) no bracketing decisions are required. The subphrase *diesel engine* has never been seen before, but *engine* has appeared as the head in two other reduced pairs: *gasoline engine* and *small engine*. Instrument and Property were each assigned once and therefore have equal weight. Both are suggested to the user in the paraphrases from section 4.4.1.

The complete phrase *repair of diesel engine* is reduced to the modifier-head-marker triple (*engine*, *repair*, *of*). The word *engine* has never been encountered as a modifier and *repair* has not occurred as a head. The preposition *of* appears in the NMR marker dictionary mapped to twelve NMRs. All twelve are suggested to the user, with *engine* and *repair* appearing in the paraphrases.

```
String (237)      the repair of diesel engines

HAIKU: Noun Modifier Relationship Analysis of current input ...

Match type 3: (diesel, _, nil) or (_, engine, nil)
[prop,inst]: 0.5

For the phrase 'diesel engine'
There is a relationship between
   (diesel) and (diesel_engine).

The NMR Analyzer's best suggestion(s) for this input:
  (1)   Property (prop): diesel_engine is diesel
  (2)   Instrument (inst): diesel is used in diesel_engine
> Please enter a number between 1 and 2
  or enter a valid NMR label ('a' to abort): 2

> For the phrase 'diesel engine'
  Do you accept the assignment:
  Instrument (inst): diesel is used in diesel_engine

   [n/a/<nmr>/Y] Y

Match type 5: nmrmDict(of,
[agt,caus,cont,equa,led,matr,obj,poss,prop,resu,src,top])

For the phrase 'repair of diesel_engine'
There is a relationship between
   (diesel_engine) and (repair).

The NMR Analyzer's best suggestion(s) for this input:
  (1)   Agent (agt): repair is performed by diesel_engine
  (2)   Cause (caus): diesel_engine causes repair
  (3)   Content (cont): diesel_engine is contained in repair
  (4)   Equative (equa): diesel_engine is also repair
  (5)   Located (led): diesel_engine is located at repair
  (6)   Material (matr): repair is made of diesel_engine
  (7)   Object (obj): diesel_engine is acted on by repair
  (8)   Possessor (poss): diesel_engine has repair
  (9)   Property (prop): repair is diesel_engine
  (10)  Result (resu): diesel_engine is a result of repair
  (11)  Source (src): diesel_engine is the source of repair
  (12)  Topic (top): repair is concerned with diesel_engine
> Please enter a number between 1 and 12
  or enter a valid NMR label ('a' to abort): 7

> For the phrase 'repair of diesel_engine'
  Do you accept the assignment:
  Object (obj): diesel_engine is acted on by repair

   [n/a/<nmr>/Y] Y

NOUN MODIFIER RELATIONSHIPS
  Object (obj): diesel_engine is acted on by repair
  Instrument (inst): diesel is used in diesel_engine

  diesel_engine isa engine
```

Figure 20: NMRA interaction for (237)

The bracketer starts with the rightmost three elements *engine repair shop* from phrase (238). Neither *engine repair* nor *engine shop* have appeared as reduced modifier-head pairs in previous bracketings,[7] so the system asks the user if *engine repair* makes sense in the context of *engine repair shop*. The user answers 'yes', providing confidence for left-branching. The bracketer moves to the left to consider *diesel engine repair.* The pair *diesel engine* has already been seen, while the pair *diesel repair* has not. Since there are no more elements to the left of *diesel*, the system brackets *(diesel engine)-repair* and expands to the right. The user has already provided the left-branching decision for *engine repair shop*, so the system brackets *((diesel engine) repair)-shop*, and then trivially, *(((diesel engine) repair) shop)*.

For NMR assignment, the compound *diesel engine* has previously been assigned Instrument, which is now suggested to the user. The reduced *engine repair* has never been seen as a compound, but the triple (engine, repair, of) at a distance of 2 has previously been assigned Object. Finally, the system asks the user to supply the NMR for the relationship between *diesel engine repair* and *diesel engine repair shop*, since *repair* has never been encountered as a modifier and *shop* has never appeared as a head.

```
String (238)        diesel engine repair shop

HAIKU: Noun Modifier Relationship Analysis of current input ...

> For the phrase 'engine repair shop'
  is that 'engine repair' [Y/n]? Y

                  __________________
                  |                |
              _______________     shop
              |             |
          ___________     repair
          |         |
      diesel     engine

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort Y
```

[7] Although *engine repair* has not appeared in bracketings, it has occurred as a modifier-head pair in *repair of engine*. Reduced pairs resulting from postmodifying prepositional phrases should also be stored to help bracketing. See section 5.3.

```
Match type 0: (diesel, engine, nil)
Most frequent NMRs: [inst] (100.0%)

> For the phrase 'diesel engine'
  Do you accept the assignment:
  Instrument (inst): diesel is used in diesel_engine

   [n/a/<nmr>/Y] Y

Match type 2: (engine, repair, _)
Most frequent NMRs: [obj] (100.0%)

> For the phrase 'diesel_engine repair'
  Do you accept the assignment:
  Object (obj): diesel_engine is acted on by diesel_engine_repair

   [n/a/<nmr>/Y] Y

Match type 7: (_, _, _)

For the phrase 'diesel_engine_repair shop'
There is a relationship between
   (diesel_engine_repair) and (diesel_engine_repair_shop).

> Please enter a valid NMR label ('a' to abort): purp

> For the phrase 'diesel_engine_repair shop'
  Do you accept the assignment:
  Purpose (purp): diesel_engine_repair_shop is meant for
                  diesel_engine_repair

   [n/a/<nmr>/Y] Y

NOUN MODIFIER RELATIONSHIPS
  Purpose (purp): diesel_engine_repair_shop is meant for
                  diesel_engine_repair
  Object (obj): diesel_engine is acted on by diesel_engine_repair
  Instrument (inst): diesel is used in diesel_engine

  diesel_engine_repair_shop isa shop
  diesel_engine_repair isa repair
  diesel_engine isa engine
```

Figure 21: NMRA interaction for (238)

The bracketer must ask the user to supply the branching decision for (239), since neither *auto repair* nor *auto centre* have appeared in bracketing before. For NMR assignment, *auto* has never appeared as a modifier, but *repair* has appeared twice as a head. Both times Object was assigned. For *auto repair centre*, *repair* has appeared once as a modifier (as Purpose in *diesel engine repair shop*); *centre* has never been encountered as a head. Purpose is suggested to the user.

```
String (239)       an auto repair centre

HAIKU: Noun Modifier Relationship Analysis of current input ...

> For the phrase 'auto repair centre'
  is that 'auto repair' [Y/n]? Y

          ______________
         |              |
    _____________      centre
   |             |
  auto         repair

> Do you accept the bracketing [r/h/m/a/Y]?
  (r)edo, (h)istory off, (m)anual, (a)bort Y

Match type 3: (auto, _, nil) or (_, repair, nil)
[obj]: 1.0

> For the phrase 'auto repair'
  Do you accept the assignment:
  Object (obj): auto is acted on by auto_repair

   [n/a/<nmr>/Y] Y

Match type 3: (repair, _, nil) or (_, centre, nil)
[purp]: 1.0

> For the phrase 'auto_repair centre'
  Do you accept the assignment:
  Purpose (purp): auto_repair_centre is meant for auto_repair

   [n/a/<nmr>/Y] Y

NOUN MODIFIER RELATIONSHIPS
  Purpose (purp): auto_repair_centre is meant for auto_repair
  Object (obj): auto is acted on by auto_repair

  auto_repair_centre isa centre
  auto_repair isa repair
```

Figure 22: NMRA interaction for (239)

4.9 Chapter Summary

HAIKU assigns NMRs to the modifier-head pairs in a noun phrase. NMRA has two main parts: the premodifier bracketing module and the NMR assignment module. The bracketer uses reduced subbracketings from previous noun phrases to bracket a new modifier sequence into modifier-head pairs. Each of these pairs is assigned an NMR by the assignment module. The assignment module also assigns NMRs to postmodifying

prepositional phrases and appositives. To decide which NMR to assign to a modifier-head-marker triple, HAIKU gathers all previous NMRs assigned to the most similar previous triples. It chooses one or more of these candidates as the most appropriate for the current triple using their weighted relative frequency. For each NMR assignment to a premodifier-head compound, HAIKU also generates an `isa` structure. In doing so it assumes that the compound is endocentric.

Performance evaluations of the bracketer showed that the system's reliance on the user for branching decisions decreases as more noun phrases are analyzed. In the *sparc* experiment, adjusting the branching frequency threshold had the expected effect: higher thresholds made the bracketer more conservative in using previous bracketing evidence. This resulted in higher accuracy, but fewer decisions were attempted by the system. In the *small engines* experiment, changing the threshold had no effect.

By the end of each experiment, HAIKU was assigning the majority of NMRs automatically, having learned from previous assignments. The system recovered between 79% and 92% of the relationships in the *small engines* text (with 95% confidence). The burden placed on the user by NMRA was low, with an average onus of 0.1 for interactions in the *small engines* experiment. No new NMRs were needed in either the *sparc* experiment or the *small engines* experiment.

5 Future Work

In this chapter I consider various related projects and extensions to the work described in the previous chapters. The first three sections deal with extensions to clause level relationship analysis (5.1), case analysis (5.2) and noun modifier relationship analysis (5.3). These extensions represent small, reasonable projects that fit within the existing TANKA framework. Crucially, HAIKU has been designed to allow many of these extensions as simple "plug-ins" that would not require significant alterations to the existing system. More significant departures appear in sections 5.4 and 5.5.

5.1 Clause Level Relationship Analysis

5.1.1 Evaluation

Clause level relationship analysis has been so far evaluated on a relatively small scale. The requirement for complete, correct parse trees up to the level of clauses limits the

amount of data available for CLR analysis.[1] The problem is compounded by the fact that data for CLRA are likely to be sparse in average texts (as already seen in *clouds* and *small engines*).

In order to test the preference rules and compare their performance to the use of stored attribute patterns, larger experiments are needed. Such experiments could also help investigate a correlation between the syntactic verb phrase features and CLRs.

5.1.2 Generalizing User Assignments

The CLR analyzer extension that learns from generalized user assignments considers four attributes: tense/modality and polarity of both clauses involved in the CLR. A more fine-grained generalization on the tense/modality attribute is possible. Two clauses may differ on their tense, but have the same level of modality (stronger, weaker, none). CLRA could replace differing tenses with a variable while maintaining the modality value if it is the same for both clauses. For example, imagine the second clause from one sentence has tense/modality of `future_simple/stronger_modal` and the second clause from another sentence has `must_present_continuous/stronger_modal`. CLRA could generalize the two attribute values to *X*`/stronger_modal`.

5.1.3 Embedded CLR Structures

The CLR analyzer cannot hold competitions between embedded CLR structures because an embedded CLR structure has no syntactic verb phrase features. Recall sentence (21) from chapter 2.

(21) The printer can print if the program issues the print command before the system shuts down.

The clause identifier `*statement1*` refers to the subtree for *the printer can print*; `*statement2*` to the subtree for *the program issues the print command* and

---

[1] I should stress that the dearth of correct parses is a weakness of the parsing technology, not of the CLRA mechanisms, although CLRA's dependence on good parsing may rightly be considered a weakness of the semantics-from-syntax philosophy.

`*statement3*` to *the system shuts down.* CLRA first assigns a CLR to the relationship between `*statement2*` and `*statement3*` resulting in CLR structure *S*. CLRA then assigns a CLR to the relationship between `*statement1*` and *S*. The tense/modality and polarity for `*statement1*` are `can_present/weak_modal` and `pos`, but the structure *S* has no distinct tense/modality or polarity values.

Similarly, the stored attribute pattern representation is not defined for embedded structures. One possibility would be to treat the attributes of the embedded structure as variables. Usually, however, variables in attribute patterns are the result of two patterns having different values for a given attribute. The variables in patterns for embedded structures would not be the result of known mismatches in attribute values and therefore may be overgeneralizations. To mitigate the effect of these variables, attribute patterns from embedded CLR structures could be given a lower weighting for pattern lookups. Because the CLRA implementation does not currently weight patterns, a general weighting scheme would also need to be invented and justified.

5.1.4 Are Attribute Patterns Empirical Preference Rules?

HAIKU uses either stored attribute patterns or preference rules to find a CLR to suggest to the user. In section 2.5.5 I said that CLR competitions giving correct analyses also result in stored patterns, so the knowledge encoded in the preference rules is not lost when using stored patterns. That is, stored attribute patterns identify what values of the verb phrase features result in the assignment of a particular CLR—knowledge that comes from the successful application of preference rules. As more sentences are analyzed, an attribute not relevant to the assignment of a CLR is more likely to be generalized to a variable than attributes that *are* relevant. For example, if the polarity of one of the clauses is not relevant in the assignment of a particular CLR (*i.e.*, may be positive or negative), the system is more likely to assign that CLR to sentences with either polarity value than if the polarity *were* relevant.

An interesting experiment would be to test this equivalence between preference rules and attribute patterns by using stored patterns in CLR competitions. In a competition, HAIKU

would look up all of the stored patterns containing the two competing CLRs (CLR_1 and CLR_2). It would then generalize all the stored patterns containing CLR_1, generalize the patterns containing CLR_2, and compare the features of the input sentence to both generalizations.[2] The better match would win.

5.2 Case Analysis

5.2.1 Assigning Cases One by One

When comparing the case marker pattern of a clause to that of a previously analyzed clause, HAIKU will suggest the previous case pattern only if both CMPs have the same number of markers. If the two CMPs have different lengths, HAIKU will not suggest any cases at all, even for markers that the CMPs have in common.

Instead of giving up on the CMP, HAIKU could suggest cases marker-by-marker. If a particular marker appears in both CMPs, the corresponding case from the previous CP could be suggested. For markers in the current CMP that do not appear in the previous CMP, HAIKU could look for other instances of the marker with the current verb and suggest those previous cases. If the marker has never appeared with the current verb, HAIKU could even check for instances of the marker with other verbs.

For example, when HAIKU is analyzing (242), it assembles the CMP `psubj-pobj-at-adv` and looks for previous CMPs with four markers. Since neither (240) nor (241) have CMPs with four markers, HAIKU asks the user to supply the case pattern.

(240) [Bernice $_{psubj/EXPR}$] lost [her keys $_{pobj/OBJ}$] [on Tuesday $_{on/TAT}$].

(241) [Charles $_{psubj/EXPR}$] lost [at the racetrack $_{at/LAT}$].

(242) [Ajax $_{psubj/????}$] lost [his poem $_{pobj/????}$] [at the club $_{at/????}$] [yesterday $_{adv/????}$].

---

[2] The generalization of modality to stronger, weaker or none as described in 5.1.2 would be less prone to overgeneralization than the more coarse generalization of tense/modality together.

Three of the four markers in the CMP for (242) have already been encountered with the verb *lose*. HAIKU could suggest Experiencer for `psubj`, Object for `pobj` and LocationAt for `at`. The user would only have to supply one case, TimeAt for `adv`.

Even when HAIKU does find a perfectly matching CMP, it asks the user to choose among all of the previous case patterns that have been associated with the CMP. Often these previous case patterns have cases in common. For example, at one point in the *clouds* experiment, twelve CPs had been accumulated for the CMP `psubj-pobj` (Table 36). If a new clause appears with the CMP `psubj-pobj`, HAIKU will suggest all twelve CPs to the user.

(1) AGT-DIR	(2) AGT-EFF	(3) AGT-OBJ
(4) CONT-EXPR	(5) EXPR-CAUS	(6) EXPR-MANR
(7) EXPR-MEAS	(8) EXPR-OBJ	(9) EXPR-TAT
(10) OBJ-MEAS	(11) RECP-MEAS	(12) RECP-OBJ

Table 36: Twelve case patterns for `psubj-pobj`

The user first looks at CP (1) considering Agent as a potential case for `psubj`. If he decides that Agent is indeed the correct case, he must choose a case for `pobj`. But now there are only really three CPs to choose from, since none of the others assign Agent to the subject of the sentence. Instead of suggesting twelve CPs for `psubj-pobj`, HAIKU could first present only the (five) cases corresponding to `psubj`. If the user chooses Agent (for example), then HAIKU could suggest only the three object-marked cases Direction, Effect and Object. The disadvantage to assigning cases one by one is that slightly more user input would be required (for example, typing a number selecting each case instead of one number for a whole CP).

5.2.2 Aggressive Case Analysis

When HAIKU encounters a clause with a CMP that has already appeared with other clauses, it suggests *all* of the previous case patterns. Although experiments have shown that the number of accumulated CPs for any CMP grows slowly, it might be possible to

reduce user burden by making HAIKU choose a single CP (or at least a proper subset of the accumulated CPs) to suggest to the user.

The choice could be based on the number of times each CP has been previously assigned, or on some score calculated from the number of times each individual case within the CPs has occurred. Experiments would be needed to see how the case analyzer's success rate is affected, and to determine if the benefit to the user of choosing from fewer suggestions is significant.

5.2.3 A General Purpose Case Pattern Dictionary

The output of case analysis is a list of the verbs in a text and the cases assigned to their arguments. Many systems that use cases for semantic analysis of text require knowledge of the kinds of case roles allowed for verbs. Briscoe & Carroll (1997) identify the need for a dictionary that contains the number and categories of a predicate's arguments. HAIKU could be used as a tool to create such a dictionary. Delisle & Szpakowicz (1997) explore the construction of a dictionary of predicate-argument structures.

By combining the marker→case associations with subcategorization information, HAIKU's output could also be used to identify which cases are core for verbs (see section 3.2.4).

5.3 Noun Modifier Relationship Analysis

5.3.1 Noun Modifier Bracketing

Storing Pairs from Prepositional Phrases

Whenever the system brackets a list of premodifiers, it stores the reduced subbracketings to help bracket subsequent noun phrases. The system could be extended to store pairs resulting from postmodifying prepositional phrases as well. For example, for the postmodifying prepositional phrase in (243) the system would store pair (244).

(243) a pot for soup

(244) (soup pot)

Storing these extra pairs would increase the likelihood of the system finding evidence when bracketing new noun phrases.

Branching Frequencies

In section 4.7.1 I noted that left-branching triples do not always outnumber right-branching triples in a text, as has been observed by others. It will be important to continue to monitor these frequencies in future experiments to determine whether the *sparc* text is unique in its predominance of right-branching triples. If left-branching predominance is confirmed, the system could be modified to guess left in the absence of other evidence.

Insensitivity to the Bracketing Threshold

Results presented in 4.7.1 also revealed the insensitivity of the bracketer to the value of the threshold applied to previous bracketing evidence. It is possible that a single technical text is unlikely to have both right-branching and left-branching evidence for a given triple, rendering the comparison threshold irrelevant. Future experiments may confirm this result or, if conflicting evidence is common, find a suitable default threshold value.

5.3.2 Other Noun Modifier Relationships

Experiments with the noun modifier relationship analyzer have not yet uncovered the need for new NMRs. Nonetheless, it is possible to think up new ones. For example, Container and Content are currently stretched to cover part-whole relationships, which might deserve their own NMRs. The possibility of a Resemblance NMR was investigated in section 4.7.3.

Another possible addition to the NMRs would be inverse relationships. An NMR and its inverse relationship express the same semantic relationship, but the roles of modifier and head are reversed. Some of the NMRs already have inverses: Container and Content; Located and Location. It seems that some other roles might have natural inverses. Source, for example, has the paraphrase "*modifier* is the source of *compound*". An inverse Source

would have the paraphrase "*compound* is the source of *modifier*", and could be introduced to account for (245).[3]

(245) *water faucet*
water_faucet is the source of *water*

Not all NMRs, however, have compelling examples for inverse relationships.

The addition of new NMRs will be left to future work. More experiments would look for noun phrases that are not covered by the existing NMRs, and provide data about the distribution of NMRs in texts. As with the cases, overrepresentation in texts might be an indication that an NMR is too general and that more specific NMRs are needed.

5.3.3 Methods for Choosing the Best NMRs

Sections 4.6.4 and 4.7.2 described various methods for choosing the best NMRs from two lists of candidates. Experimental evaluation of the different methods was inconclusive. Future experiments should continue to evaluate all of the implemented methods to determine if one method is significantly better than the others. Alternatively, a more theoretically sound calculation for choosing NMRs could be sought.

5.3.4 Taxonomic Information from NMRs

In section 4.7.3 I noted that not all NMRs should result in the generation of `isa` structures. At present the only such NMR is Property, for which `isa` generation is disabled. If new NMRs are added to HAIKU, it is possible that there would be other relationships (such as a Resemblance relationship) that should not result in `isa` structures either. The simplest solution would be to add such relationships to the list of `isa`-defeating relationships along with Property.

A more general solution would be to associate with each NMR what type of hierarchical structure should be automatically generated for that NMR. In this way, any taxonomic relationship could be associated with individual NMRs. The existing NMRs would all

---

[3] Without an inverse Source NMR, (245) would be assigned Product.

indicate an `isa` relationship except Property, which would indicate that no taxonomic relationship should be generated. Other taxonomic relationships could be indicated as well, such as the `islikea` relationship from section 4.7.3 or even a meronymic relationship for NMRs like Instrument or part-whole NMRs.

5.4 A Unified Set of Semantic Relationships

The choice of a surface representation of a sentence is often arbitrary, as illustrated by (246) and (247), both of which describe the same events.

(246) A man was murdered yesterday with a handgun because a jealous husband returned early.

(247) the murder of a man yesterday with a handgun because of the early return of a jealous husband

In both examples, a handgun is the Instrument of a murdering event, a jealous husband is the Agent of a returning event, etc. There is obviously some overlap in HAIKU's three types of semantic relationships.

Deverbal nouns such as *murder* and *return* are known to mark case-like relationships. Levi (1978) calls such nouns nominalizations, and accounts for their semantics with four case-like roles: Act, Agent, Patient, Product. HAIKU has NMRs that resemble cases to account for compounds with deverbal noun heads.

Just as some nouns participate in case-like relationships with their modifiers, stative verbs are more appropriately analyzed by assigning NMRs between their arguments. Frawley (1992) argues that there is no thematic role in (248) between *ball* and *is*, or between *red* and *is*. Rather, there is a relationship between *ball* and *red* directly, while the verb *is* is irrelevant.

(248) The ball is red.

By unifying HAIKU's three sets of semantic relationships into one set, it might be possible to mix and match evidence from the different levels. Lauer (1995a: 154) writes: "to interpret *garbage collection*, knowledge of the semantics of, and case roles for, the verb *collect* are needed." HAIKU may indeed have knowledge of the case roles for *collect* accumulated during case analysis. It is possible to uncover the verb root of a deverbal noun (Hull & Gomez 1996; Barker *et al.* 1997). The NMR analyzer could check stored case patterns for that verb as evidence for semantic relationship assignment.

Unifying HAIKU's semantic relationships would be a large undertaking and there is little precedent for such an all-encompassing set. In particular, validation would be a considerable endeavour.

5.5 Other Directions Altogether

The extensions described in sections 5.1, 5.2 and 5.3 could all be implemented within the existing HAIKU framework. In this section I investigate projects that represent a significant departure from TANKA principles.

5.5.1 Fully Automatic Recognition of Semantic Relationships

Semi-automatic analysis in HAIKU proceeds linearly through a text to avoid disorienting the user. Sentences appearing later in a text, however, might have constructions that could be used to disambiguate constructions appearing earlier. A fully automatic analyzer could process sentences and phrases in any order to arrive at a cumulative analysis of a complete text. The system could proceed in several passes (iteratively), making all of the unambiguous decisions and interpretations first and storing them. In subsequent passes more constructions could be unambiguously analyzed using structures from previous passes. The number of passes could be limited by requiring that some minimum number of relationships be recovered on each pass for processing to continue.

Several issues would require investigation in building a fully-automatic HAIKU. Would it be justifiable, for example, to use stored structures from texts already analyzed to help

start the processing of a new text? Could existing knowledge resources such as WordNet replace HAIKU's user as oracle? What would be the effect of incorrect parse trees assumed to be correct? What would be the effect of incorrect stored semantic structures on future analyses?

5.5.2 Extending HAIKU with WordNet

HAIKU avoids semantic information associated with nouns, verbs and adjectives since such open-ended information would require a large knowledge engineering effort. Some such knowledge is available, however, in WordNet. HAIKU could make use of WordNet in several places.[4]

In clause level relationship analysis the WordNet synonym sets or hypernym sets for the verbs of each of the clauses could be added to the stored attribute patterns. There may be a correlation between certain kinds of verbs (such as *cause* and *prevent*) and CLRs. Patterns with a non-nil intersection of synonym (or hypernym) sets in the verb attribute would be considered matching. It might also be possible to discover modal expressions in clauses using WordNet. Synonym sets for adverbs in a clause could be expanded until one of a small number of known modal adverbs is found (such as *certainly*, *probably*, *possibly*).

In case analysis, case marker pattern matching could be extended to match on semantics of the case fillers. For example, if two syntactic subjects from two different sentences share a hypernym, they may mark the same case.

In sections 4.4.1 and 4.7.3 I proposed using WordNet pertainyms to improve awkward noun modifier relationship paraphrases. HAIKU's NMR analyzer could also generalize NMR arguments using WordNet. Instead of looking for exact matches on modifiers and heads, HAIKU could look for intersection of their synonym sets or hypernym sets. For example, if the modifier of the current triple has no match in the stored triples, the system

---

[4] Delisle (1996) describes a system that allows information from WordNet to be added to DIPETT's dictionary.

could look for matches using elements of the modifier's synonym set. If no previous triples match elements of the synonym set, the system could look for matches using elements of the modifier's hypernym set.

5.5.3 Other Sources of Syntactic Information

The experiments described in this dissertation underlined HAIKU's reliance on good parse trees from DIPETT. It would be an interesting project to adjust the system to work with other sources of syntactic information. In its current form HAIKU depends on the actual structures produced by DIPETT, but it could be modified to accept structures from other parsers, as long as they provide comparable information. For example, any parser that identifies syntactic subjects, objects and prepositional phrases attached to a verb could feed case analysis in HAIKU. Any system that identifies noun premodifiers (such as the parsers described by Voutilainen & Padró 1997 or Zhai 1997) could feed NMR analysis.

6 Conclusions

The thesis that I have set out to defend in this dissertation has been that semantic relationships in sentences form a model of a text and that it is possible to acquire this model interactively. There are a number of ways to make the user-assisted acquisition relatively painless: define general, domain-independent semantic relationships; find clues in syntax that hint at underlying semantic relationships; ask the user simple, well-directed questions when consultation is necessary; apply simple learning techniques to approved analyses to allow the system to act more autonomously on future sentences.

6.1 Summary

The bulk of this dissertation was about the design and implementation of parts of TANKA's semantic analyzer. The purpose of HAIKU is to recognize semantic relationships

in the sentences of a text. Chapters 2, 3 and 4 dealt with HAIKU's three semantic analysis modules.

6.1.1 Clause Level Relationship Analysis

In chapter 2 I described the clause level relationship analyzer. CLRs are assigned to clauses in coordinate, subordinate and correlative syntactic relationships. The CLR marker dictionary lists a subset of candidate CLRs appropriate to the clausal connective in a sentence. HAIKU holds competitions between the candidates using preference rules that choose between pairs of CLRs. The rules prefer one CLR over another depending on the connective polarity and the verb phrase polarity, tense and modality. The CLR with the most points at the end of all competitions is presented to the user for approval. To help with future assignments, approved CLRs and clausal attributes are stored and generalized on unequal attribute values.

6.1.2 Case Relationships

Since case analysis was previously described by Delisle (1994), in chapter 3 I showed in detail the process of choosing a set of semantic relationships. Initial selection comes from a broad review of existing systems in theoretical and computational linguistics, seeding the set with linguistically and practically motivated relationships. Building a marker dictionary grounds the set in the surface-syntactic phenomena that express the relationships. Marker dictionary construction also exposes weaknesses in the coverage of the set.

In chapter 3 I also described a methodical evaluation of the cases using their distribution in texts and among the case markers. The evaluation criteria were *generality*, *completeness* and *uniqueness* (absence of *redundant* and *superfluous* cases).

6.1.3 Noun Modifier Relationship Analysis

The noun modifier relationship analyzer described in chapter 4 assigns NMRs to each of the modifiers of a noun. Premodifiers must be bracketed into local modifier-head pairs, each of which received its own NMR. The semi-automatic bracketer handles adjective

and noun premodifier sequences of any length. It learns from reduced subbracketings of previous phrases to decide whether a subphrase is left-branching or right-branching.

HAIKU assigns NMRs to modifier-head pairs from bracketed premodifier sequences and from postmodifying prepositional phrases and appositives. Modifier-head-marker triples are compared to previous triples. NMRs associated with the most similar such previous triples are candidates for the relationship between the modifier and head under analysis. One of these candidates is chosen as the best NMR based on its frequency of occurrence in previous assignments. The NMR analyzer also generates automatically taxonomic relationships between a compound and its head.

6.2 Goals Revisited

In chapter 1 I presented five goals for the project. I will now return to those goals and relate them to the results presented in the preceding chapters.

6.2.1 Semantic Relationships

The first goal was to construct usable, general purpose sets of semantic relationships. I have described the construction of three such sets: clause level relationships, cases and noun modifier relationships. Each of the lists was inspired by a union of semantic relationships found in research in discourse analysis, in case theory and valency theory and on noun phrases.

The clause level relationships cover the senses of a large number of coordinators and subordinators found in conventional and electronic dictionaries. In the *building code*, *clouds* and *small engines* experiments no new CLRs were needed to account for the semantic relationships between events represented by clauses in those texts.

HAIKU's case marker dictionary contains several hundred lexical items (atomic and phrasal markers) whose many senses are covered by the cases. Some cases are marked by fewer markers than others, but no case exists for only a single marker. Inspection of the

representation of each case in the marker dictionary identified potentially over-general or over-specific cases.

The existing case set has also been sufficiently complete to account for verb argument relationships in many texts, including (most recently) the *clouds* text and the *small engines* text. On the other hand, several cases were not assigned in those experiments. In chapter 3 I rejected the possibility that cases such as Recipient, TimeFrom and TimeTo are superfluous, even if they appear to occur rarely in the test texts. The inclusion of similar cases in many other case lists is indirect evidence of their utility.

Semantic relationships within noun phrases in the *sparc* and *small engines* texts were all covered by the existing NMRs. Nonetheless, there are possibilities for new relationships (see section 5.3.2). Experiments with more texts will determine if the coverage of the current set of NMRs is sufficient.

6.2.2 Semi-Automatic Recognition of Semantic Relationships

The second goal from chapter 1 was the completion of the HAIKU semantic analyzer. Chapters 2, 3 and 4 contain details of the design and implementation of each of its modules. As planned, the components make use of linguistic knowledge and of small amounts of closed, domain-independent knowledge. They learn from user assignments to make better suggestions to the user. The success of the implementation is supported by the data from experiments, as summarized in the following sections.

6.2.3 Learning

The third goal was to evaluate HAIKU's ability to learn from previous analyses to make better suggestions to the user. Results of the experiments described in chapters 2, 3 and 4 show that as more text is analyzed, the proportion of correct analyses by the system increases steadily.

For the *clouds* and *small engines* texts, the clause level relationship analyzer got 76% of CLR assignments correct using preference rules only. The problem with the preference rules is that if they prefer the wrong CLR for a given input, they will always prefer the

wrong CLR for similar inputs. Experience using HAIKU has revealed that correcting the system on the same mistakes is simple, but annoying.

Learning from user assignments helps HAIKU avoid such behaviour, but there is the problem of knowing when to use stored CLR attribute patterns and when to stay with the preference rules. For the 127 CLR assignments from the *clouds* and *small engines* texts, CLRA got 77% correct using stored attribute patterns and preference rules. More experiments are needed to determine if there is a significant difference in performance between the two methods and also to see if performance improves over time when using stored attribute patterns.

HAIKU assigned 439 case patterns in the *clouds* experiment and 584 patterns in the *small engines* experiment. For both experiments the proportion of clauses for which the correct CP was among the system's suggestions increased over the course of the experiment. For the *clouds* experiment HAIKU had correctly assigned 50% of the CPs after 208 clauses. It correctly assigned 72% of the CPs for the rest of the text. In the *small engines* experiment HAIKU had correctly assigned half of the CPs after the first 100 clauses. For the rest of the clauses, it correctly assigned 68%. Figure 4 and Figure 5 in chapter 3 illustrate how the cumulative number of correct assignments by HAIKU increased at a higher rate than user assignments throughout both experiments.

In the *sparc* experiment HAIKU's bracketer made 63% of the 194 bracketing decisions correctly: 52% in the first half of the experiment, 74% in the second half. For the *small engines* experiment the bracketer made 62% of the 164 decisions correctly. For the first quarter of that experiment the number hovered just over 50% after which it jumped fairly quickly up to about 65% where it stayed for the rest of the experiment. Always guessing left branching would have resulted in 91% success for the *small engines* text, but only 45% for the *sparc* text. Based on others' observations of branching frequencies, it is unlikely that either of these two texts is representative of texts in general. Yet the *sparc* text shows that the predominance of left-branching compounds is not universal.

The noun modifier relationship analyzer assigned 72% of the NMRs in the *sparc* experiment correctly: 51% in the first third of the experiment, 83% for the rest of the text. The *small engines* experiment showed similar numbers: 49% of the first 100 assignments were made correctly by the system, 71% of the assignments for the rest of the text. On average NMRA made 69% of the assignments correctly over the entire text. The improvements show that NMRA can learn from user assignments. Furthermore, since only 57% of the *sparc* assignments and 43% of the *small engines* assignments were perfect matches on reduced modifier-head pairs, partial matching was responsible for a significant portion (15% and 26% respectively) of the correct analyses.

6.2.4 Coverage

The fourth goal was to evaluate HAIKU's coverage of semantic relationships in a text and its sensitivity to errors in parse trees and missing parse trees. The system can recover many semantic relationships even from incorrect or fragmentary parse trees.

It was no surprise to discover that clause level relationship analysis suffered the most from parse errors in the experiments. HAIKU got the opportunity to assign CLRs for 67% of the clause-clause constructions in the *clouds* text. In the more syntactically complex *small engines* text, only 28% of constructions requiring CLRs were analyzed.

Case analysis is the least sensitive to parse errors because it allows the user to correct mistakes in case marker patterns. Sampling 100 verbs at random from the *small engines* text showed that HAIKU extracted between 92% and 99% of the case patterns in the text (at 95% confidence). The number would have been much smaller if the system were limited to assigning case patterns to correct CMPs only. In the *clouds* experiment, 69% of the CMPs constructed from the parse trees were correct. For the *small engines* experiment, only 55% of CMPs were correct.

Noun modifier relationship analysis works with the smallest parse tree fragments and should be the least sensitive to parse errors. With DIPETT's partial parsing facility, many noun phrases are parsed correctly even if the parser cannot connect their parses to larger trees. Unlike in case analysis, however, the user is not given the opportunity to repair

incorrect parses. Sampling 100 modifier-noun pairs in the *small engines* text showed that between 79% and 92% of the pairs in the text received NMRs (at 95% confidence).

6.2.5 User Burden

The final goal stated in chapter 1 was to evaluate the burden that semi-automatic HAIKU analysis places on the user. Experiments have shown that the system can be used to recover multiple semantic relationships from several hundred sentences in just a few days (starting from scratch) without overly taxing the user.

Burden is evaluated in terms of the onus ratings (see section 1.4.4) assigned to each interaction during the *clouds* and *small engines* experiments. The onus ratings assigned for interactions in both experiments appear in Table 37. In those experiments 1739 of the 1981 HAIKU interactions (88%) were considered trivial, while only 27 of the interactions (1%) were considered difficult.

		onus				
		0	*1*	*2*	*3*	*average*
clouds	*CLRA*	50	1	0	0	0.02
	CA	384	50	5	0	0.14
small engines	*CLRA*	17	4	0	0	0.19
	CA	480	89	15	0	0.20
	NMRA	808	71	7	0	0.10

Table 37: Onus ratings for the clouds *and* small engines *experiments*

Burden can also be evaluated by looking at the number of HAIKU interactions per sentence and the amount of time spent on each sentence. In the *clouds* experiment there were on average 0.10 CLRs and 0.86 case patterns assigned in each sentence, requiring on average 1 minute and 49 seconds. In the *small engines* experiment there were 0.04 CLRs, 1.05 case patterns and 1.59 NMRs per sentence, coincidentally also requiring 1 minute 49 seconds. In both experiments the average time per sentence relative to the number of HAIKU interactions decreased over the course of the experiment.

6.3 Closing Words

Semantic relationships in sentences form a model of a text that can be acquired interactively, with relatively little burden on the user. This thesis is defended through the satisfaction of five goals: the construction of sets of general, portable semantic relationships; the implementation of a semi-automatic system that recognizes the semantic relationships in sentences using syntactic and linguistic clues; confirmation of the system's ability to learn to work more autonomously; confirmation that the semantic relationships are sufficient to model complete texts; and confirmation that the burden placed on the user is low. The design, implementation and evaluation of HAIKU directly addressed each of the goals.

The conclusion is that the interactive recovery of semantic relationships in a text is viable. The syntax does hold clues about the underlying semantic relationships between syntactic elements in a sentence. HAIKU can use these clues to recognize a good portion of the semantic relationships in a text automatically, By learning from mistakes corrected by a user it can improve its performance. Finally, there are ways to elicit knowledge from the user without requiring open-ended subjective knowledge entry.

Systems that acquire deep semantic knowledge from texts fully automatically are not forthcoming. This dissertation has shown that it is feasible to acquire some knowledge from texts starting from scratch—without big semantic knowledge bases, without big tagged corpora, with just a little help from a user and a little learning.

7 References

ANDERSON, JOHN M. (1971). *The Grammar of Case: towards a localist theory.* Cambridge: Cambridge University Press.

ATKINSON, HENRY F. (1990). *Mechanics of Small Engines.* New York: McGraw-Hill, Inc.

BACH, EMMON & ROBERT T. HARMS, eds. (1968). *Universals in Linguistic Theory.* New York: Rinehart and Winston.

BÄCKLUND, INGEGERD (1984). *Conjunction-Headed Abbreviated Clauses in English.* Stockholm: Almqvist & Wiksell, International.

BARKER, KEN (1994). "Clause-Level Relationship Analysis in the TANKA System." TR-94-07, Department of Computer Science, University of Ottawa.
`http://www.site.uottawa.ca/~kbarker/papers/TR-94-07.ps`

BARKER, KEN (1996). "The Assessment of Semantic Case Roles Using English Positional, Prepositional and Adverbial Case Markers." TR-96-08, Department of Computer Science, University of Ottawa.
`http://www.site.uottawa.ca/~kbarker/papers/TR-96-08.ps`

BARKER, KEN (1997). "Noun Modifier Relationship Analysis in the TANKA System". TR-97-02, Department of Computer Science, University of Ottawa.
`http://www.site.uottawa.ca/~kbarker/papers/TR-97-02.ps`

BARKER, KEN (1998). "A Trainable Bracketer for Noun Modifiers." The *Twelfth Canadian Conference on Artificial Intelligence*, Vancouver, 196-210.

BARKER, KEN & SYLVAIN DELISLE (1996). "Experimental Validation of a Semi-Automatic Text Analyzer." TR-96-01, Department of Computer Science, University of Ottawa.
`http://www.site.uottawa.ca/~kbarker/papers/TR-96-01.ps`

BARKER, KEN & STAN SZPAKOWICZ (1995). "Interactive Semantic analysis of Clause-Level Relationships." *Proceedings of the Second Conference of the Pacific Association for Computational Linguistics*, Brisbane, 22-30.

BARKER, KEN & STAN SZPAKOWICZ (1998). "Semi-Automatic Recognition of Noun Modifier Relationships." *Proceedings of COLING-ACL '98*, Montréal.

BARKER, KEN, TERRY COPECK, SYLVAIN DELISLE & STAN SZPAKOWICZ (1997). "Systematic Construction of a Versatile Case System". *Journal of Natural Language Engineering* 3(4), December, 1997.

BARKER, KEN, SYLVAIN DELISLE & STAN SZPAKOWICZ (1998). "Test-Driving TANKA: Evaluating a Semi-Automatic System of Text Analysis for Knowledge Acquisition." The *Twelfth Canadian Conference on Artificial Intelligence*, Vancouver, 60-71.

BAUER, LAURIE (1983). *English Word-Formation.* Cambridge: Cambridge University Press.

BRISCOE, TED & JOHN CARROLL (1997). "Automatic Extraction of Subcategorization from Corpora." *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C.

BRUCE, BERTRAM (1975). "Case systems for natural language." *Artificial Intelligence* 6 (1975), 327-360.

CAMPE, PETRA (1994). *Case, Semantic Roles, and Grammatical Relations: a Comprehensive Bibliography.* Amsterdam: John Benjamins.

CHAN, LOREN B. (1989). *SPARCstation 1 Installation Guide.* Sun Microsystems, Inc.

COATES, JENNIFER (1983). *The Semantics of the Modal Auxiliaries.* London: Croom Helm.

COLE, RONALD A., JOSEPH MARIANI, HANS USZKOREIT, ANNIE ZAENEN & VICTOR ZUE, eds. (1996). *Survey of the State of the Art in Human Language Technology.*
`http://www.cse.ogi.edu/CSLU/HLTSurvey/`

COOK, WALTER A., S.J. (1979). *Case Grammar: Development of the Matrix Model (1970-1978).* Washington: Georgetown University Press.

COOK, WALTER A., S.J. (1989). *Case Grammar Theory.* Washington: Georgetown University Press.

COPECK, TERRY, KEN BARKER, SYLVAIN DELISLE & STAN SZPAKOWICZ (1997). "What is Technical Text?" *Language Sciences* 19(4), 391-424.

DAHLGREN, KATHLEEN (1988). *Naïve Semantics for Natural Language Understanding.* Boston: Kluwer Academic Publishers.

DELISLE, SYLVAIN (1994). "Text processing without A-Priori Domain Knowledge: Semi-Automatic Linguistic analysis for Incremental Knowledge Acquisition." Ph.D. thesis, TR-94-02, Department of Computer Science, University of Ottawa.

DELISLE, SYLVAIN (1996). "Le Traitement Automatique du Langage Naturel au Service de l'Ingénieur de la Connaissance: le Système READER." *Actes de la Conférence Internationale sur le Traitement Automatique des Langues et ses Applications Industrielles (TAL+AI 1996)*, Moncton, 60-66.

DELISLE, SYLVAIN & STAN SZPAKOWICZ (1995). "Realistic Parsing: Practical Solutions of Difficult Problems." *Proceedings of the Second Conference of the Pacific Association for Computational Linguistics*, Brisbane, 59-68.

DELISLE, SYLVAIN & STAN SZPAKOWICZ (1997). "Extraction of Predicate-Argument Structures from Texts." *Recent Advances in Natural Language Processing (RANLP 1997)*, Bulgaria, 318-323.

DELISLE, SYLVAIN, TERRY COPECK, STAN SZPAKOWICZ & KEN BARKER (1993). "Pattern Matching for case analysis: A Computational Definition of Closeness." *Proceedings of ICCI-93*, 310-315.

DELISLE, SYLVAIN, KEN BARKER, TERRY COPECK & STAN SZPAKOWICZ (1996). "Interactive Semantic analysis of Technical Texts." *Computational Intelligence* 12(2), May, 1996, 273-306.

VAN DIJK, TEUN A. (1977). *Text and Context: Explorations in the Semantics and Pragmatics of Discourse.* London: Longman.

DOWNING, PAMELA A. (1977). "On the Creation and Use of English Compounds." *Language* 53:810-842

FABRE, CÉCILE (1996). "Interpretation of Nominal Compounds: Combining Domain-Independent and Domain-Specific Information." *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-96)*, Copenhagen, 364-369.

FILLMORE, CHARLES J. (1968). "The Case for Case." in Bach & Harms (1968).

FININ, TIMOTHY W. (1986). "Constraining the Interpretation of Nominal Compounds in a Limited Context." in Grishman & Kittredge (1986: 163-173).

FOWLER, HENRY W. (1984). *A Dictionary of Modern English Usage: Second Edition* (revised by Sir Ernest Gowers). Oxford: Oxford University Press.

FRANKS, BRADLEY (1995). "Sense Generation: A 'Quasi-Classical' Approach to Concepts and Concept Combination." *Cognitive Science* 19:441-505.

FRAWLEY, WILLIAM. (1992). *Linguistic Semantics.* Hillsdale: Lawrence Erlbaum Associates, Publishers.

GEORGE, STEFFI (1987). *On "Nominal Non-Predicating" Adjectives in English.* Frankfurt am Main: Peter Lang.

GRISHMAN, RALPH (1995). *MUC-6.*
`http://cs.nyu.edu/cs/faculty/grishman/muc6.html`

GRISHMAN, RALPH & RICHARD KITTREDGE, eds. (1986). *Analyzing Language in Restricted Domains: Sublanguage Description and Processing.* Hillsdale: Lawrence Erlbaum.

GRISHMAN RALPH & BETH SUNDHEIM (1996). "Message Understanding Conference - 6: A Brief History." *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-96)*, Copenhagen, 466-471.

GOMEZ, FERNANDO (1995). "Acquiring Intersentential Explanatory Connections in Expository Texts." *International Journal of Human-Computer Studies* 44(1), 19-44.

HALLIDAY, M.A.K. & RUQAIYA HASAN (1976). *Cohesion in English.* London: Longman.

HERMERÉN, LARS (1978). *On Modality in English: A Study of the Semantics of the Modals.* Lund: CWK Gleerup.

HIRSCHMAN, LYNETTE & HENRY S. THOMPSON (1996). "Overview of Evaluation in Speech and Natural Language Processing." in Cole *et al.* (1996).

HOBBS, JERRY (1983). "Why is Discourse Coherent?" in Neubauer (1983).

HOVY, EDUARD H. (1993). "Automated Discourse Generation Using Discourse Structure Relations." *Artificial Intelligence* 63:1-2 (October, 1993), 341-385.

HULL, RICHARD D. & FERNANDO GOMEZ (1996). "Semantic Interpretation of Nominalizations." *Proceedings of AAAI-96*, Portland, 1062-1068.

JENSEN, KAREN, GEORGE E. HEIDORN & STEPHEN D RICHARDSON, eds. (1993). *Natural Language Processing: The PLNLP Approach.* Boston: Kluwer Academic Publishers.

KAMP, HANS (1975). "Two Theories about Adjectives." in Keenan (1975).

KEENAN, EDWARD L., ed. (1975). *Formal Semantics of Natural Language: Papers from a Colloquium Sponsored by the King's College Research Centre, Cambridge.* Cambridge: Cambridge University Press.

KEHLER, ANDREW (1993a). "The Effect of Establishing Coherence in Ellipsis and Anaphora Resolution." *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics.* Columbus, Ohio, 62-69.

KEHLER, ANDREW (1993b). (personal communication).

KEMPEN, GERARD, ed. (1986). *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics.* Dordrecht: Martinus Nijhoff Publishers.

KNOTT, ALISTAIR & ROBERT DALE (1994). "Using Linguistic Phenomena to Motivate a Set of Rhetorical Relations." *Discourse Processes* 18(1), 35-62.

KNOTT, ALISTAIR & CHRIS MELLISH (1996). "A Feature-Based Account of the Relations Signalled by Sentence and Clause Connectives." *Language and Speech* 39, 143-183.

LARRICK, NANCY (1961). *Junior Science Book of Rain, Hail, Sleet & Snow.* Champaign: Garrard Publishing Company.

LARSON, MILDRED L. (1984). *Meaning-Based Translation: A Guide to Cross-Language Equivalence.* Lanham: University Press of America.

LASCARIDES, ALEX, NICHOLAS ASHER & JON OBERLANDER (1992). "Inferring Discourse Relations in Context." *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics.* Newark, Delaware, 1-8.

LAUER, MARK (1995a). "Designing Statistical Language Learners: Experiments on Noun Compounds." Ph.D. thesis, Macquarie University.

LAUER, MARK (1995b). "Corpus Statistics Meet the Noun Compound: Some Empirical Results." *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics.* Cambridge. 47-54.

LAUER, MARK & MARK DRAS (1994). "A Probabilistic Model of Compound Nouns." *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence.* Armidale. 474-481.

LEHMANN, SABINE, STEPHAN OEPEN, SYLVIE REGNIER-PROST, KLAUS NETTER, VERONIKA LUX, JUDITH KLEIN, KIRSTEN FALKEDAL, FREDERIK FOUVRY, DOMINIQUE ESTIVAL, EVA DAUPHIN, HERVÉ COMPAGNION, JUDITH BAUR, LORNA BALKAN & DOUG ARNOLD (1996). "TSNLP — Test Suites for Natural Language Processing."

Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-96), Copenhagen, 711-716.

LEONARD, ROSEMARY (1984). *The Interpretation of English Noun Sequences on the Computer*. Amsterdam: North-Holland.

LEVI, JUDITH N. (1978). *The Syntax and Semantics of Complex Nominals*. New York: Academic Press.

LEVIN, BETH (1993). *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago: The University of Chicago Press.

LIBERMAN, MARK & RICHARD SPROAT (1992). "Stress and Structure of Modified Noun Phrases." *Lexical Matters (CSLI Lecture Notes, 24)*. Stanford: Center for the Study of Language and Information.

LYONS, JOHN (1995). *Linguistic Semantics: An Introduction*. Cambridge: Cambridge University Press.

MANN, WILLIAM C. & SANDRA A. THOMPSON (1986a). "Relational Propositions in Discourse." *Discourse Processes* 9:1 (1986), 57-90.

MANN, WILLIAM C. & SANDRA A. THOMPSON (1986b). "Rhetorical Structure Theory: Description and Construction of Text Structures." in Kempen (1986).

MANN, WILLIAM C. & SANDRA A. THOMPSON (1988). "Rhetorical Structure Theory: Toward a functional theory of text organization." *Text* 8:3 (1988), 243-281.

MILLER, GEORGE A., ed. (1990). "WordNet: An On-Line Lexical Database." *International Journal of Lexicography* 3(4).

MUC-6 (1996). *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann.

NEUBAUER, FRITZ, ed. (1983). *Coherence in Natural Language Texts*. Hamburg: Buske.

OFLAZER, KEMAL & YILMAZ, OKAN (1996). "A Constraint-Based Case Frame Lexicon." *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-96)*, Copenhagen, 854-859.

ONTARIO MINISTRY OF HOUSING (1991). "The Ontario Building Code." *Ontario Regulation* 413/90. Queen's Printer for Ontario.

PALMER, F. R. (1979). *Modality and the English Modals*. London: Longman.

PUSTEJOVSKY, JAMES & FEDERICA BUSA (1995). *A Revised Template Description for Time (v3)*.
`http://cs.nyu.edu/cs/faculty/grishman/time-guidelines.v3_1.html`

PUSTEJOVSKY, JAMES, S. BERGLER & P. ANICK (1993). "Lexical Semantic Techniques for Corpus Analysis." *Computational Linguistics* 19(2). 331-358.

QUIRK, RANDOLPH, SIDNEY GREENBAUM, GEOFFREY LEECH & JAN SVARTVIK (1985). *A Comprehensive Grammar of the English Language.* London: Longman.

RESNIK, PHILIP STUART (1993). "Selection and Information: A Class-Based Approach to Lexical Relationships." Ph.D. thesis, IRCS Report 93-42, University of Pennsylvania.

SANDERS, TED J.M., WILBERT P.M. SPOOREN & LEO G.M. NOORDMAN (1992). "Toward a Taxonomy of Coherence Relations." *Discourse Processes* 15 (1992), 1-35.

SCHANK, ROGER C. (1975). *Conceptual Information Processing.* Amsterdam: North-Holland Publishing Company.

SCHANK, ROGER C. & R.P. ABELSON (1977). *Scripts, Plans, Goals and Understanding.* Hillsdale, NJ: Erlbaum.

SCHIFFRIN, DEBORAH (1987). *Discourse Markers.* Cambridge: Cambridge University Press.

SINCLAIR, JOHN, ed. (1991). *Collins COBUILD English language Dictionary.* London: Collins Publishers.

SLATOR, BRIAN M., SHAHRZAD AMIRSOLEYMANI, SANDRA ANDERSEN, KENT BRAATEN, JOHN DAVIS, RHONDA FICEK, HOSSEIN HAKIMZADEH, LESTER MCCANN, JOSEPH RAJKUMAR, SAM THANGIAH & DANIEL THUREEN (1990). "Towards Empirically Derived Semantic Classes." *Proceedings of the Fifth Rocky Mountain Conference on Artificial Intelligence*, Las Cruces, 257-262.

SOMERS, H.L. (1987). *Valency and case in Computational Linguistics.* Edinburgh: Edinburgh University Press.

SOWA, JOHN F. (1984). *Conceptual Structures: Information Processing in Mind and Machine.* Reading: Addison-Wesley Publishing Company.

SPARCK JONES, KAREN (1994). "Towards Better NLP System Evaluation." *Proceedings of the Human Language Technology Workshop*, San Francisco: Morgan Kaufmann, 102-107.

SPARCK JONES, KAREN & BRAN BOGURAEV (1987). "A Note on the Study of cases." *Computational Linguistics* 13(1-2), 65-68.

SPARCK JONES, KAREN & JULIA R. GALLIERS (1996). *Evaluating Natural Language Processing Systems: An Analysis and Review. (Lecture Notes in Artificial Intelligence 1083)*, New York: Springer-Verlag.

TER STAL, WILCO (1996). "Automated Interpretation of Nominal Compounds in a Technical Domain." Ph.D. thesis, University of Twente, The Netherlands.

STEIN, JESS, ed. (1983). *The Random House Dictionary of the English language*. New York: Random House.

SUMMERS, DELLA, ed. (1987). *Longman Dictionary of Contemporary English: New Edition*. Essex: Longman.

VAN VALIN, ROBERT D., ed. (1993). *Advances in Role and Reference Grammar*. Amsterdam: John Benjamins.

VANDERWENDE, LUCY (1993). "SENS: The System for Evaluating Noun Sequences." in Jensen *et al.* (1993: 161-173).

VELARDI, P., M.T. PAZIENZI & M. FASOLO (1991). "How to Encode Semantic Knowledge: a Method for Meaning Representation and Computer-Aided Acquisition." *Computational Linguistics* 17(2), 153-170.

VOUTILAINEN, ATRO & LLUÍS PADRÓ (1997). "Developing a Hybrid NP Parser." *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington.

WARREN, BEATRICE (1978). *Semantic Patterns of Noun-Noun Compounds*. Göteborg: Acta Universitatis Gothoburgensis.

WARREN, BEATRICE (1984). *Classifying Adjectives*. Göteborg: Acta Universitatis Gothoburgensis.

WILKS, YORICK A., BRIAN M. SLATOR & LOUISE M. GUTHRIE. (1996). *Electric Words: Dictionaries, Computers, and Meanings*. Cambridge: The MIT Press.

WU, DEKAI (1993). "An Image-Schematic system of Thematic Roles." *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, Vancouver, 323-332.

ZHAI, CHENGXIANG (1997). "Fast Statistical Parsing of Noun Phrases for Document Indexing." *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C.

Appendix I: CLR Marker Dictionary

```
/*************************************************************************/
/* file: HAIKU_clrmDICT.PL                                               */
/* version: August 12, 1997                                              */
/* author: Ken Barker                                                    */
/*************************************************************************/

:- assert(haiku_version(12,8,97)).

/*************************************************************************/
/* clrmDict(connective, connective polarity,                             */
/*         list of potential CLRs with associated example sentence       */
/*         and 'direction' flag (1 = syntactic arg1 is CLR arg1          */
/*                               2 = syntactic arg2 is CLR arg1)         */
/*                                                                       */
/* In general, for coordinators the args are unordered.  For             */
/* subordinators, the main clause is syntactic arg1 and the              */
/* subordinate clause is arg2.                                           */
/* In example sentences the CLR arg1 is capitalized, NOT the             */
/* syntactic arg1.                                                       */
/* Ex:                                                                   */
/*  clrmDict(because,pos,                                                */
/*           [enab(2,'The printer works BECAUSE THE POWER IS ON.')]).*/
/*************************************************************************/
```

```
clrmdict(after,pos,
    [prec(2,'We came AFTER THEY LEFT.'),
     enab(2,'The printer can print AFTER THE PAPER TRAY IS FILLED')]).
clrmdict(also,pos,
    [conj(1,'He was mean, ALSO he was ugly.')]).
clrmdict(although,neg,
    [detr(2,'He wouldn''t come ALTHOUGH I ASKED HIM.')]).
clrmdict(and,pos,
    [conj(1,'He ate AND she drank.'),
     prec(1,'SHE READ FOR AN HOUR AND she went to bed.')]).
clrmdict(as,pos,
    [ctmp(1,'We were leaving AS they arrived.'),
     caus(2,'AS IT WAS GETTING LATE, we adjourned.'),
     enab(2,'The printer can print AS IT HAS POWER.'),
     entl(2,'AS THE FILE WAS PRINTED, the job must have run.'),
     prev(2,'AS THE JOB FAILED, the file was not printed.')]).
clrmdict(because,pos,
    [caus(2,'The program died BECAUSE THE MACHINE HAD NO MORE MEMORY.'),
     entl(2,'The job must have been killed BECAUSE THE LOG SHOWS IT.'),
     enab(2,'The printer should work BECAUSE THE POWER IS ON.'),
     prev(2,'The file did not print BECAUSE THE JOB FAILED.')]).
clrmdict(before,pos,
    [prec(1,'YOU''D BETTER BE GONE BEFORE I come.'),
     enab(1,'YOU MUST TURN ON THE COMPUTER BEFORE you can use it.')]).
clrmdict(but,neg,
    [prev(2,'We would attend BUT WE HAVE NO MONEY.'),
     enab(2,'The printer would not have printed BUT THE POWER WAS ON.'),
     detr(1,'WE HAVE NO MONEY BUT we will attend.')]).
clrmdict(considering,pos,
    [detr(2,'CONSIDERING SHE ARRIVED ONLY RECENTLY she is well
adjusted.'),
     prev(2,'CONSIDERING I HAVE NO MONEY I will be unable to attend.')]).
clrmdict('either-or',neg,
    [disj(1,'EITHER I will do homework OR I will watch television.'),
     prev(1,'EITHER give yourself up OR I will shoot.')]).
clrmdict(except,neg,
    [enab(2,'It would not have printed EXCEPT THE PROGRAM SUCCEEDED.'),
     prev(2,'We would attend EXCEPT WE HAVE NO MONEY.')]).
clrmdict(even_if,pos,
    [detr(2,'We will not attend EVEN IF WE ARE INVITED.')]).
clrmdict(for,pos,
    [caus(2,'The file did not print FOR THE JOB FAILED.'),
     enab(2,'The printer can print FOR IT HAS POWER.'),
     entl(2,'The file must have printed FOR THE JOB SUCCEEDED.'),
     prev(2,'The file must not have printed FOR THE JOB FAILED.')]).
clrmdict(if,pos,
    [prev(2,'IF THE POWER IS OFF, the printer won''t work.'),
     enab(2,'IF THE POWER IS ON, the printer can be used.'),
     entl(2,'IF THE LIGHT IS ON, he must be home.')]).
...
```

Appendix II: Case Marker Dictionary

```
/*************************************************************************/
/* file: HAIKU_cmDICT.PL                                                 */
/* version: March 5, 1993                                                */
/* author:  Ken Barker                                                   */
/*************************************************************************/

:- assert(haiku_version(5,3,93)).

/*************************************************************************/
/* cmDict("CM word",                                                     */
/*         "list of potential Cases with associated example sentence     */
/*          and 'commonness' flag (1 = common, 2 = uncommon)")           */
/*                                                                       */
/* Ex: cmDict(aboard,                                                    */
/*             [lat(2,'He climbed ABOARD THE SHIP.')]).                  */
/*************************************************************************/

/*            Prepositional and Positional Case Markers                  */

cmDict(aboard,
   [lat(2,'He climbed ABOARD THE SHIP.')]).
cmDict(about,
   [cont(1,'He wrote ABOUT MATHEMATICS.'),
    ltru(2,'He wandered ABOUT THE CITY this afternoon.'),
    lat(2,'The flowers were arranged ABOUT A SINGLE WHITE ROSE.')]).
cmDict(above,
   [lat(1,'Put the horseshoe ABOVE THE MANTLE.'),
    dir(1,'Look ABOVE THE CLOUDS.')]).
```

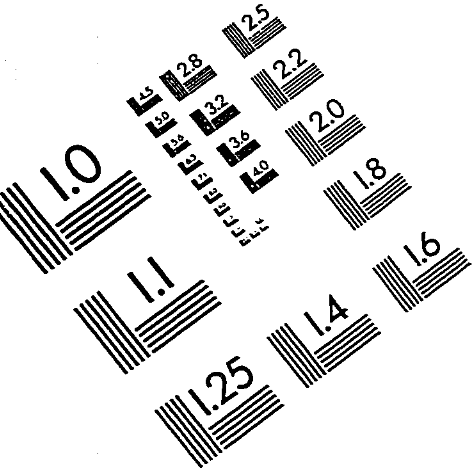
```
cmDict(across,
   [lat(1,'I live in the house ACROSS THE RIVER.'),
    dir(1,'We walked ACROSS THE STREET to see.')]).
cmDict(after,
   [ord(1,'File it AFTER THE ALPHABETIC ENTRIES.'),
    tat(1,'File it AFTER FIVE.'),
    lat(1,'He drove AFTER THE BUS.')]).
cmDict(against,
   [opp(1,'She spoke AGAINST THE PROPOSAL.'),
    dir(1,'The rain beats AGAINST THE WINDOW.'),
    lat(1,'Mount the picture AGAINST A BLUE BACKGROUND.'),
    purp(2,'He is saving AGAINST RETIREMENT.')]).
cmDict(ahead_of,
   [lat(1,'He stood AHEAD OF ALL THE OTHER PLAYERS.'),
    ord(1,'She finished AHEAD OF THE REST OF THE CLASS.')]).
cmDict(along,
   [lat(1,'We stood ALONG THE BANK as the boat went by.'),
    ltru(1,'He ran ALONG THE RIVER to the bridge.')]).
cmDict(alongside,
   [lat(2,'The tug came ALONGSIDE THE LINER.')]).
cmDict(amid,
   [lat(2,'He stood AMID THE GUESTS at the party.')]).
cmDict(amidst,
   [lat(2,'He stood AMIDST THE GUESTS at the party.')]).
cmDict(among,
   [lat(1,'Stand AMONG US or you''ll be volunteered.'),
    acmp(1,'Divide it AMONG THEM.')]).
cmDict(amongst,
   [lat(1,'Stand AMONGST US or you''ll be chosen.'),
    acmp(1,'Divide it AMONGST THEM.')]).
cmDict(apart_from,
   [excl(1,'APART FROM THAT, what is left to do?')]).
cmDict(around,
   [ltru(1,'We walked AROUND THE COURTYARD sightseeing.')]).
cmDict(as,
   [manr(1,'She acted AS MY AGENT in Europe last year.')]).
cmDict(as_of,
   [tfrm(2,'The system is up AS OF YESTERDAY.')]).
cmDict(aside_from,
   [excl(2,'ASIDE FROM THAT, what is left to do?')]).
cmDict(at,
   [dir(1,'The deer ran right AT THE HUNTERS.'),
    lat(1,'I stood AT THE DOOR.'),
    tat(1,'The convict will be executed AT HIGH NOON.'),
    manr(1,'The car moves AT HIGH SPEED.'),
    cont(1,'She is good AT ARTS.'),
    meas(1,'It stopped AT FIFTY.'),
    caus(1,'She was amazed AT HIS GALL.')]).
cmDict(atop,
   [lat(2,'Place the book ATOP THE BUREAU.')]).
cmDict(bar,
   [excl(2,'He is the best BAR NONE.')]).
cmDict(barring,
   [excl(2,'He is the best BARRING NONE.')]).
cmDict(before,
   [ord(1,'This goes BEFORE THAT.'),
    tat(1,'This was BEFORE WW II.'),
    lat(2,'Stand BEFORE ME.')]).
...
```

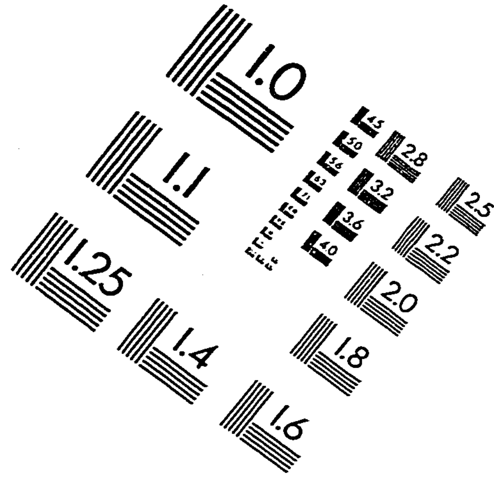
Appendix III: NMR Marker Dictionary

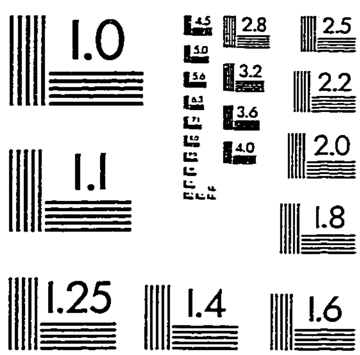
```
/*************************************************************************/
/* file: HAIKU_nmrmDICT.PL                                               */
/* version: July 3, 1997                                                 */
/* author: Blazej Szpakowicz                                             */
/*************************************************************************/

:- assert(haiku_version(3,7,97)).

/*************************************************************************/
/* nmrmDict(marker, list of potential NMRS)                              */
/*                                                                       */
/* Ex: nmrmDict(for,                                                     */
/*              [benf('printer for students'),                           */
/*               purp('printer for theses')]).                           */
/*************************************************************************/

nmrmDict(about,
   [loc('flowers about a gravestone'),
    top('book about Kennedy')]).
nmrmDict(according_to,
   [caus('order according_to importance'),
    src('the gospel according_to Mark')]).
nmrmDict(across,
   [loc('trip across the lake')]).
nmrmDict(after,
   [loc('the file after the A''s'),
    time('life after death')]).
```

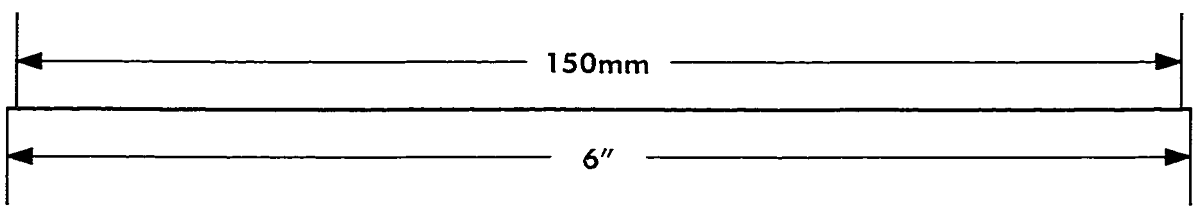
```
nmrmDict(against,
    [loc('back against the wall'),
     dest('travel against the wind'),
     obj('protection against the cold'),
     top('protest against the government')]).
nmrmDict(ahead_of,
    [loc('a position ahead_of one''s rivals'),
     time('arrival ahead_of expectations')]).
nmrmDict(aimed_at,
    [dest('arrow aimed_at a target')]).
nmrmDict(along,
    [loc('a walk along the river'),
     top('research along certain lines')]).
nmrmDict(amid,
    [loc('a man amid the crowd'),
     time('peace amid the fighting')]).
nmrmDict(amidst,
    [loc('land amidst the water'),
     time('a battle amidst a war')]).
nmrmDict(among,
    [agt('action among the community'),
     loc('cat among the pigeons'),
     src('a view among critics')]).
nmrmDict(amongst,
    [agt('a movement amongst the natives'),
     loc('weeds amongst the flowers'),
     src('belief amongst the people')]).
nmrmDict(at,
    [caus('anger at his reply'),
     dest('shot at a foe'),
     loc('house at the border'),
     stat('country at war'),
     time('meal at night'),
     top('master at chess')]).
nmrmDict(based_on,
    [caus('friendship based_on money'),
     src('film based_on a book')]).
nmrmDict(because_of,
    [caus('acquittal because_of a technicality')]).
nmrmDict(behind,
    [loc('garden behind a house')]).
nmrmDict(beside,
    [loc('ditch beside the road')]).
nmrmDict(between,
    [agt('war between countries'),
     loc('a passageway between rooms'),
     poss('secret between friends')]).
nmrmDict(beyond,
    [loc('a building beyond a field'),
     prop('courage beyond measure')]).
nmrmDict(by,
    [agt('assault by the military'),
     caus('death by asphyxiation'),
     inst('trial by jury'),
     loc('tree by the fence'),
     src('a film by Spielberg')]).
nmrmDict(concerning,
    [top('a pamphlet concerning elections')]).
...
```

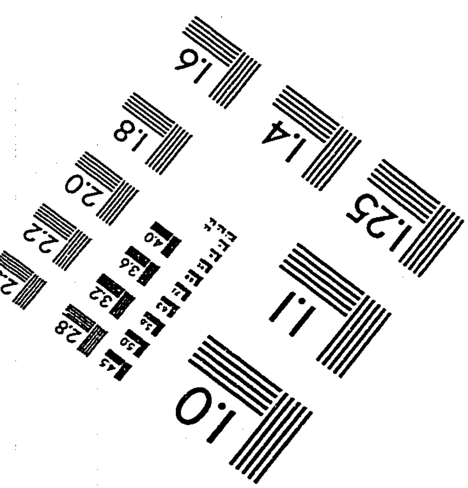
IMAGE EVALUATION TEST TARGET (QA-3)











APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989