

<b>2</b>	<b>Clause Level Relationship Analysis</b>	<b>15</b>
<b>2.1</b>	<b>Introduction</b>	<b>15</b>
2.1.1	Semantic Relationships	16
2.1.2	Tense and Modality	18
<b>2.2</b>	<b>Input Structures</b>	<b>19</b>
2.2.1	Verb Sequence Features	19
2.2.2	Clausal Organization	19
	Coordination or Subordination	19
	Correlative Coordination	20
	Subordinator/Conjunct Correlation	20
<b>2.3</b>	<b>The Clause Level Relationships</b>	<b>20</b>
2.3.1	CLR Glossary	22
	Causal CLRs	22
	Temporal CLRs	23
	Conjunctive CLRs	23
<b>2.4</b>	<b>The CLR Marker Dictionary</b>	<b>24</b>
<b>2.5</b>	<b>Assigning CLRs</b>	<b>25</b>
2.5.1	Verb Phrase Polarity and Connective Polarity	25
2.5.2	Verb Phrase Tense and Modality	27
	Absence of Modals	28
	Modals in Negative Verb Phrases	28
	Marginal Modals	29
	Should as a Past Tense of Shall	29
2.5.3	Using Argument Features to Choose CLRs	30
	CLR Preference Rules	31
2.5.4	CLR Competitions	35
2.5.5	Using User Assignments to Choose CLRs	36
	CLR Assignment Attributes	36
	When to Use User Assignments	37
	Least-General Generalization of Attribute Patterns	38
	Avoiding Overgeneralization	39
<b>2.6</b>	<b>Evaluation</b>	<b>39</b>
2.6.1	Diagnostic Evaluation	39
2.6.2	Performance Evaluation	40
	System Performance	40
	Coverage	42
2.6.3	Can CLRA Learn?	42
<b>2.7</b>	<b>An Example</b>	<b>43</b>
<b>2.8</b>	<b>Chapter Summary</b>	<b>46</b>

## 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 CLR’s (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 CLR’s, 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 CLR’s 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.

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.

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)

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 causes E2  
E1 makes E2 occur or exist. E1 is sufficient to cause E2  
and the occurrence or existence of E1 is required.

(10) *[The file printed <sub>E2</sub>] because [the program issued a print command <sub>E1</sub>].* (CAUS)

##### *Detraction (DETR)*

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

(11) *Although [the server was very busy <sub>E1</sub>], [the program ran <sub>E2</sub>].* (DETR)

##### *Enablement (ENAB)*

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

(12) *[The printer can print <sub>E2</sub>] if [the paper tray contains paper <sub>E1</sub>].* (ENAB)

##### *Entailment (ENTL)*

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

(13) *[The printer will print <sub>E2</sub>] if [a print command is issued <sub>E1</sub>].* (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<sub>E2</sub>] since [the hard disk crashed<sub>E1</sub>].* (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<sub>E1</sub>] until [I changed the toner cartridge<sub>E2</sub>].* (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).

The next step was to map each marker sense to the appropriate CLRs. These marker→CLR mappings were determined by studying the conventional dictionary<sup>1</sup> 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



### 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.

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

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].*

## 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<sub>E2</sub>] if the paper tray is empty.* (PREV)

(23) *[The document will print<sub>E2</sub>] 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<sub>E2</sub>] unless the paper tray is empty.* (PREV)

(25) *[The document will not print<sub>E2</sub>] 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.<sup>2</sup> 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

---

<sup>2</sup> 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.

relationships as sufficient or insufficient suggests that a binary division into stronger modals and weaker modals is appropriate for disambiguating the CLR's.

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

### ***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.<sup>3</sup> 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.*

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).

---

<sup>3</sup> 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.

(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.<sup>4</sup> 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.

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.

---

<sup>4</sup> 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 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<sub>E2</sub>] because there was sufficient memory.* (CAUS)

(46) *[The program may not work<sub>E2</sub>] because memory is low.* (DETR)

*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<sub>E2</sub>] because the machine was out of memory.* (CAUS)

(48) *[The program should run<sub>E2</sub>] 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<sub>E2</sub>].* (CAUS)

(50) *The program failed, therefore [the system will crash<sub>E2</sub>].* (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

*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.

only condition a) of the polarity reversal rule will apply since Causation is never marked by a negative connective.

(51) *[We adjourned the meeting  $E_2$ ] as it was getting late.* (CAUS)

(52) *[We could not continue the meeting  $E_2$ ] 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  $E_2$ ].* (DETR)

(54) *If the power is on, [the printer can quickly print files  $E_2$ ].* (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  $E_2$ ].* (DETR)

(56) *If the system responds, [the computer must be functioning  $E_2$ ].* (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  $E_2$ ].* (DETR)

(58) *If the system is hung, [the computer will not work  $E_2$ ].* (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 <sub>E2</sub>].* (ENAB)

(62) *If the program responds, [the computer must be working <sub>E2</sub>].* (ENTL)

*Enablement vs. Prevention*

These two CLR's 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 <sub>E2</sub>].* (ENAB)

(64) *If the power is off, [the printer will not print <sub>E2</sub>].* (PREV)

*Entailment vs. Prevention*

Although Entailment and Prevention are both sufficient causal CLR's, 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 <sub>E2</sub>].* (+E2, +conn: ENTL)

(66) *If the printer is out of paper, [the file will not print <sub>E2</sub>].* (-E2, +conn: PREV)

(67) *Unless the printer has paper, [the file will not print <sub>E2</sub>].* (-E2, -conn: ENTL)

(68) *Unless the printer is out of paper, [the file will print <sub>E2</sub>].* (+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 <sub>E1</sub>]  
[they can be used to detect the slightest wear on engine parts <sub>E2</sub>]*

The positive connective *since* is listed in the CLR marker dictionary mapped to five CLRs: Causation, Enablement, Entailment, Prevention and Precedence. HAIKU holds ten

<i>CLR1</i>	<i>CLR2</i>	<i>Winner</i>	<i>Reason</i>
Causation	Enablement	Enablement	the modal <i>can</i> appears in E2
Causation	Entailment	Entailment	the modal <i>can</i> 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 <i>can</i> 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)

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.

	<i>wins</i>	<i>losses</i>	<i>ties</i>	<i>points</i>
<i>Causation</i>	2	2	0	4
<i>Enablement</i>	<b>4</b>	<b>0</b>	<b>0</b>	<b>8</b>
<i>Entailment</i>	3	1	0	6
<i>Precedence</i>	0	4	0	0
<i>Prevention</i>	1	3	0	2

Table 9: Points accumulated by each CLR in 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.

#### *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:

<i>Connective</i>	<i>T1</i>	<i>P1</i>	<i>T2</i>	<i>P2</i>	<i>CLR</i>
-------------------	-----------	-----------	-----------	-----------	------------

*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.<sup>5</sup> 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*

---

<sup>5</sup> 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.

*generalization* (defined below) of the current pattern with each of the most closely matching stored patterns.

### *Assigning CLR's Based on Stored Attribute Patterns*

For a new input, the system gathers all stored attribute patterns that have all five attributes matching the input.<sup>6</sup> It then builds a list of the CLR's associated with the stored patterns and chooses the most frequent CLR's 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.

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  $\mathbf{V}$  be the attribute pattern to be stored for the most recent CLR assignment. HAIKU stores  $\mathbf{V}$  as-is with its assigned CLR. If  $\mathbf{V}$ '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  $\mathbf{V}$  than  $\mathbf{V}'$ .

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

HAIKU computes the lgg of  $\mathbf{V}$  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	ent1

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
---	------	-----------------------------	-----	----------------------------	-----	------

<sup>6</sup> Generalized attributes perfectly match any value.

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  $\mathbf{V}$  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  $\mathbf{V}$  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.

<i>accept</i>	<i>choose</i>	<i>supply</i>	<i>reorder</i>
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.

	<i>accept</i>	<i>choose</i>	<i>supply</i>	<i>reorder</i>
<i>original (aggressive) experiment</i>	35 (69%)	2 (4%)	14 (27%)	7
<i>aggressive retest</i>	37 (73%)	2 (4%)	12 (23%)	0
<i>conservative retest</i>	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.

	<i>accept</i>	<i>choose</i>	<i>supply</i>
<i>original experiment (21 CLR)s</i>	14 (67%)	2 (10%)	5 (24%)
<i>complete retest (76 CLR)s</i>	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.<sup>7</sup> The small number of CLR's 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.<sup>8</sup>

For the *clouds* experiment, the number of CLR's 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

---

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

<sup>8</sup> 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).

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

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

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.<sup>9</sup> 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
----	-----------------------------	-----	----------------------------	-----	------

<sup>9</sup> Recall that for the marker *if* the first syntactic argument (main clause) is the second CLR argument (see section 2.4).

```

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 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*.<sup>10</sup> 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	<b>enab</b>
----	-----------------------------	-----	----------------------------	-----	-------------

<sup>10</sup> Preference rule competitions would have favoured Entailment for (83) because the strong modal and positive polarity in E2.

```

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)

## 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.*