Interactive Semantic Analysis of Technical Texts

SYLVAIN DELISLE
Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Trois-Rivières, Québec, Canada G9A 5H7
email: Sylvain_Delisle@uqtr.uquebec.ca

KEN BARKER, TERRY COPECK, STAN SZPAKOWICZ
Department of Computer Science, University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
email: {kbarker, terry, szpak}@csi.uottawa.ca

Abstract

Sentence syntax is the basis for organizing semantic relations in TANKA, a project that aims to acquire knowledge from technical text. Other hallmarks include an absence of precoded domain-specific knowledge; significant use of public-domain generic linguistic information sources; involvement of the user as a judge and source of expertise; and learning from the meaning representations produced during processing. These elements shape the realization of the TANKA project: implementing a trainable text processing system to propose correct semantic interpretations to the user. A three-level model of sentence semantics including a comprehensive Case system provides the framework for TANKA's representations. Text is first processed by the DIPETT parser, which can handle a wide variety of unedited sentences. The semantic analysis module HAIKU then semi-automatically extracts semantic patterns from the parse trees and composes them into domain knowledge representations. HAIKU's dictionaries and main algorithm are described with the aid of examples and traces of user interaction. Encouraging experimental results are described and evaluated.

KEYWORDS: Knowledge Acquisition, Text Analysis, Semantic Networks, Parsing, Semantic Analysis, Case Analysis

1 Introduction

1.1 Motivation

Knowledge acquisition from text has only recently become an active research pursuit. To date most large-scale knowledge acquisition (KA) efforts have relied on structured interviews or repertory grids filled out by domain experts. A new idea is to tap the vast amount of knowledge contained in texts, especially technical texts. This may involve searching large linguistic and domain knowledge bases (dictionaries, semantic networks and so forth) for representations of the meaning of a text fragment at hand. Few machine-readable domain knowledge bases exist. A first goal then is to create them, and a system to *extract* representation elements from the text cannot presume they exist.

The goal of the TANKA project (Text ANalysis for Knowledge Acquisition) is to build a conceptual model of the domain described in a real text using as little *a priori* domain knowledge as possible. In the absence of such knowledge, the system makes use of publicly available lexical information sources and the surface-syntactic features of the text to propose analyses to a user for approval. Reliance on a user during processing offers several advantages over dependence on knowledge coded in advance. First, the amount of work required of the TANKA user will not exceed what a knowledge engineer would do to build an equivalent knowledge base ahead of time: our user supplies information *only as needed* to process the text at hand. Second, TANKA then lessens this burden by making informed *suggestions* to which the user often need only assent. This is easier for the user than volunteering a semantic interpretation out of the blue. Finally, our system *improves* in this role: as it processes more and more text, it learns from previous analyses and user interactions to make better suggestions. The user will, however, remain in the loop indefinitely. The *deliberate* scarcity of semantic information means that the system will ask the user to approve its conclusions and to resolve the many ambiguities that are beyond its capacity. This suits our belief that it is important to validate input before it is added to a knowledge base.

The remainder of this section presents the TANKA system. Section 2 introduces the framework for Case Analysis in TANKA and explains how it is applied to a variety of sentences found in technical texts. Section 3 presents the algorithm for interactive Case Analysis with rote learning used in TANKA's HAIKU semantic analysis module. An extensive literature review follows in Section 4. Section 5 gives a thorough discussion of planned work.

1.2 The TANKA System

TANKA's text analysis system starts by applying syntactic and surface-level lexical knowledge to perform surface-syntactic immediate-constituent analysis of an unsimplified real-world text. This is followed by semantic analysis to determine the relationships between these constituents. The resulting semantic interpretation is then transformed into a small self-contained semantic network and merged with a larger incrementally-growing network that represents the system's current model of the domain described by the text. Research will determine how easy it is to extend a model by processing additional texts on the same domain.

Data modelling is suited to this sort of text-to-concepts transformation. An Entity-Relationship model of an enterprise could be derived semi-automatically from a text describing enterprise operations in terms of objects and processes/activities. Completeness of the model would depend on completeness of the text. Another application is machine learning from text which incorporates examples (Matwin & Szpakowicz 1992, 1993; Delisle *et al*. 1994): semantic structures of the examples as well as the domain theory in the text body are translated into Horn clauses, on which an Explanation-based Learning (EBL) engine performs symbol-level learning.

TANKA is implemented in Quintus Prolog 3.1 on Sun workstations—DIPETT (see 1.2.1) and HAIKU (see 1.2.2) are implemented both in Quintus and in SISCtus 2.1/#9 Prolog. The organization of the system is shown in Figure 1.

1.2.1 Syntactic Analysis

Syntactic analysis is done by DIPETT (Domain Independent Parser of English Technical Texts), a broad-coverage Definite Clause Grammar parser (Delisle and Szpakowicz 1991; Copeck *et al.* 1992; Delisle 1994) whose rules are based primarily on Quirk *et al.* (1985). In the absence of precoded semantic knowledge, these general (domain-independent) syntactic rules are the only precoded knowledge used in TANKA.

DIPETT produces a single initial parse. This may be more productive than approaches in which multiple parse trees must be dealt with subsequently in semantic analysis (Brown and Nirenburg 1990); see also Blanchon (1992) where disambiguation requires asking the user to select an interpretation from a list of possibilities. DIPETT's "best first" parse tree, however, is not always correct. Prepositional phrase misattachment is relatively frequent, and legitimate but unlikely parses may be caused by lexical noun-verb ambiguity. Since misparses affect semantic analysis, we plan to add an interactive module soon in which the user can manipulate text

fragments to correct any misparsing. It will ensure that the parse is correct before continuing with semantic analysis (see section 5.2.2).

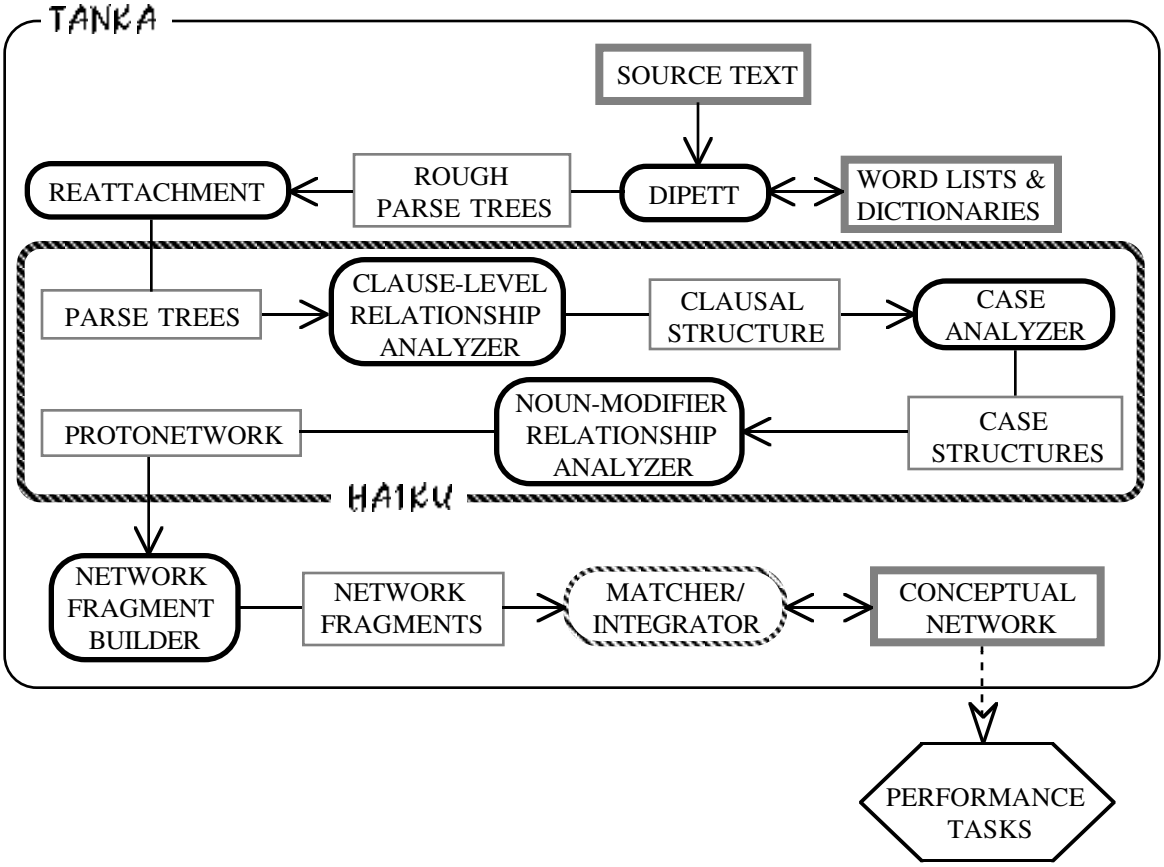

Figure 1. Overview of the TANKA System.

Anaphora is another issue that must be dealt with before semantic analysis can proceed. Unresolved pronominal references in particular make semantic analysis difficult unless holes are permitted in the acquired domain model. Subtle anaphora resolution using knowledge-rich techniques is beyond the capacity of our system. For the purpose of knowledge acquisition we can make do with resolution of the simpler types of anaphora that occur most frequently.

We have again chosen an interactive semi-automatic approach to the resolution of pronoun anaphora[1] in DIPETT. Pronoun resolution is done between parsing and semantic analysis by a procedure that operates on parse trees. It finds non-pronominal referents for personal pronouns.

---

1 More precisely, our resolution process works both for anaphora (the referent may be found in the current or the previous input) and a restricted case of cataphora (the referent may be found only in the current input). Input may be a sentence, a clause or a text fragment.

The method can be described as interactive "explicitization". The user is shown a list of syntactically suitable noun phrases and a pronoun is replaced with the noun phrase which is selected.

1.2.2 Semantic Analysis

The HAIKU semantic analysis module performs semantic processing on three levels: Clause-Level Relationships, Cases and Noun-Modifier Relationships.

Clause-Level Relationships (CLRs) are semantic relationships between acts, events or states realized in text by finite clauses connected by conjunctions or conjunctive punctuation. CLR Analysis assigns a semantic label to the link between pairs of such clauses connected in the text. The CLR Analyzer, fully implemented, is documented elsewhere (Barker 1994). Phenomena analyzed by this module at the sentence level are not described in this paper, which focuses on the Case Analyzer. The interpretation rules in Section 2 deal only with Case semantics, that is, the verb + argument level.

Cases represent semantic relationships between the main verb in a clause and its syntactic arguments (subject, objects, prepositional phrases and adverbials). The links labelled by Cases correspond to roles in the activity—for example, the Agent Case identifies who acts. Cases appear in text as predicate-argument structures, in which each Case is denoted by a marker and occupied by a particular filler—for example, in "The thief broke the window with a stone", "a stone" is the Instrument by which the action is performed. Section 3 presents the algorithms and data structures used in the Case Analyzer's processing, which is guided by the system but supervised by the user. Full details of HAIKU's Case Analyzer are presented in Delisle (1994).

Noun-Modifier Relationships (NMRs) represent semantic relationships between the head noun in a phrase and its modifiers (adjectives, nouns, relative clauses and so on). The relationships identified between a noun and its modifiers will eventually be used to construct links between objects and properties**,** and between objects and other objects in the growing conceptual network. Research on NMRs and NMR Analysis is in progress.

1.2.3 Conceptual Network Processing

The structures resulting from semantic analysis capture relationships between concepts presented in the text. These structures will be used to build network fragments: concepts become nodes and the semantic relationships become labelled, directed arcs between them. These network fragments will be integrated into a growing network that is intended to be a model of the domain represented by the text. Aspects of the Conceptual Network representation used in

TANKA are discussed in Szpakowicz & Koperczak (1990) and Yang & Szpakowicz (1991a, 1991b, 1994).

2 Case Analysis in HAIKU

This section explains precisely what Case Analysis (CA) means and what sorts of clauses and phrases DIPETT can parse which HAIKU must therefore deal with. We then present an original framework for semantic analysis geared towards knowledge acquisition *via* text processing. This framework underlies the current version of HAIKU. Section 3 continues by presenting high-level algorithmic aspects of the Case Analysis process and some illustrative examples.

2.1 Case Theory

Case Analysis in HAIKU is based on Case Theory (Fillmore 1968; Somers 1987), which focuses on the simple clause and on the verb within it. Cases capture that part of the meaning of a constituent[2] of the clause conveyed by its relationship with the verb. A Case expresses the semantic role of the argument denoted by a constituent in the situation denoted by a verb. Cases other than location or manner are normally unique in the clause (but it could be argued that, repeated locations of a situation add up to a single complex location).

Cases are semantic abstractions. Their appearance in a text is signalled by *Case markers.* These lexico-syntactic clues are either *lexical* (for example, a preposition that introduces a prepositional phrase) or *positional* (subject, direct object, indirect object). In Fillmore's classical example *John opened the door with the key* the noun phrase *John* denotes the positional Agent Case, the noun phrase *the door* denotes the positional Object Case, and the prepositional phrase *with the key* denotes the Instrument Case (lexical, introduced by *with*).

2.2 Limitations of Simple Case-based Frameworks

2.2.1 Limitations

Case Theory has its problems. First, selecting the set of names and semantic relations to use is a delicate issue. We will return to this in Section 2.3.1. Second, traditional Case theory requires that Cases and arguments in any proposition match one-to-one (Fillmore 1968). This

---

[2] Traditionally, Case Theory focuses on semantic roles (Cases) that noun phrases play with verbs. In order to handle complex sentences, noun equivalents and nominal clauses of any sort can be the object of Cases.

seems to be violated for verbs like *sell*, which has its Agent and Source filled by the same argument. Third, systems based on Case Theory or Case grammars ignore the relationships between clauses. Finally, Case systems also ignore relationships like possession (for example, *the baby's pyjamas*) which operate directly between nouns and do not involve the verb.

Violation of the one-Case-per-argument constraint is not a significant problem for us: it happens infrequently and several solutions to the problem have been proposed, for instance Somers (1987: ch. 8).

A significant amount of a text's semantic information is conveyed by relationships among its clauses. Causal relationships are often realized by connected clauses, as in: *Jim is deemed a resident of Canada for income tax purposes because he is serving abroad in the armed forces.* HAIKU responds to this by treating the semantic relationships between clauses separately in its Clause-Level Relationship Analyzer.

Natural language is expressive enough to let the same semantic concept be represented by vastly different syntactic structures. The sentence *My program created several large files on the server last Tuesday* can be analyzed to account for the relationships between *created* and its arguments. Case Analysis would not, however, be able to account for the phrase *the creation of several large files on the server last Tuesday by my program* because it lacks a verb. We are currently studying and classifying the semantic relationships that can exist between a noun and its modifiers—adjectives, possessives, prepositional phrases, relative clauses, other nouns and so on. This would extend our semantic model to the full range of syntactic phenomena.

2.2.2 The Need for an Extended Theory

Although Case systems have always been controversial (see Somers 1987: chapters 7-9), we have opted to use Case Analysis in TANKA because it accounts well for the surface syntactic form of text fragments[3]. This form is all that is given in the knowledge-scant acquisition task we have set ourselves. We wish, however, to formulate a semantic theory that accounts for realistic sentences. Such sentences tend to be more complicated than ones to which Case systems are typically applied, in which a single main verb is related in a straightforward way to a few simple

---

[3] Wu (1993) discusses a thematic role system which does *not* closely match the surface syntactic form. More precisely: "The proposed representation depends more heavily on nested structures than ordinary case systems, thereby reducing the number of thematic roles. A side effect of this is that surface cases do not map straightforwardly onto thematic roles in a one-to-one fashion. For instance, the sentence `(5) John moved his furniture from San Francisco to Berkeley` is analyzed as an outer *volitive* frame whose EFFECT role is filled by an inner *locative*." (Wu, 1993: 327).

noun phrases (NPs) and prepositional phrases (PPs). Because we want to represent the meaning of entire texts, we need an *extended* theory that will deal with the clause-level phenomena usually neglected by simple Case systems.

HAIKU will encounter a variety of syntactic phenomena, some quite complex. Traditional Case theory has little to say about many of them. Here is a list of syntactic structures that establishes the scope of the semantic analysis required to meet TANKA's objectives. Section 2.3.3 describes how these structures will be interpreted in the full semantic framework we propose.

— **complex NPs and embedded clauses:**
NPs with post-modifiers such as a relative clause, PP, or appositive ("Bob, *the owner*, ...");
nominal clauses such as to_infinitive_clause, ing_clause, that_clause ("*That Bob did it* is clear");
comparative clauses ("Bob is *more intelligent than your sister*").

— **stative and conjoined verbs:**
stative verbs ("Bob *is* on holiday");
conjoined main verbs ("Bob *washed and dried* the dishes").

— **structures beyond simple clauses:**
complex and compound sentences ("*When Bob saw it, he spoke up*");
sentences with structures other than Subject-Verb-Object and declarative clauses: Here+be sentences, ("*Here is the newest version of the compiler*"), existential-There sentences, ("*There are three people in Bob's family*") and imperatives ("*Print your business letters on the laser printer*").

— **conjoined structures:**
inter-sentential conjunctions ("Bob presented his arguments persuasively. *However* no one changed their minds.");
intra-sentential conjunctions: NPs and nominal clauses, PPs, verb phrases (VPs), subordinate clauses (sentence-level modifiers), adverbial and adjectival phrases.

2.3 A Comprehensive Framework for Case Analysis

2.3.1 The Set of Cases

HAIKU uses the set of 28 Cases[4] presented in Table I. This set was developed (Barker *et al.* 1993) during several phases of work in which we considered, analyzed and criticized extant Case lists published by the following authors: Bruce (1975), Celce-Murcia (1975), Cook (1979), Fillmore (1968), Grimes (1975), Larson (1984), Loewen (1988), Somers (1987), and Sparck Jones and Boguraev (1987).

The 28 Cases are grouped into five categories on the basis of common characteristics shared by particular relationships between the act and a Case Filler. Though not explicitly used by the TANKA system, these categories help the user choose the right Case during interaction. The *Participant* category consists of Cases whose entities are directly involved in the act. The *Causality* group includes relationships with events or entities assisting or deterring the act. The relationship may or may not be sufficient to cause or prevent the act. *Time* and *Space* Case relationships place the act at an absolute or relative position in time or space. Finally, *Quality* categorizes other varied Case relationships between a verb and its arguments.

A perennial question is what grain to employ in a Case system. Domain-specific Cases are efficacious but difficult to port from one domain to another. Universal Cases on the other hand tend to carry insufficient meaning. We have chosen them notwithstanding, and plan to make our general Cases more meaningful by associating typical semantic categories with the fillers for each (see section 5.2.3).

2.3.2 Notation

i) Major syntactic groups are denoted by the usual abbreviations (NP for noun phrase, NC for nominal clause). Subscripts indicate a group's syntactic role (NP_{subj} for a NP serving as subject, PP_{compl} for a complement PP).

ii) Cases associated with a verb are shown thus:

$$\mathbf{CA}(S) \rightarrowtail$$

$$[\,Case_1 : LABEL_1(\, sgr_1 : string_1, mv\,), ..., Case_n : LABEL_n(\, sgr_n : string_n, mv\,)]$$

---

[4] Case names themselves are not so important. What really matters is that TANKA and its users refer to them in a consistent fashion. For a proposal on how to handle Cases without using names (that is, labels), see Delisle (1990). A related topic is that of semantic primitives; for a review of related work and an interesting discussion, see Fass (1989).

"**CA**(S) ➵ ..." means "Case Analysis of sentence S produces this set of Cases ...". The expression "$Case_i$: $LABEL_i$(sgr_i : $string_i$, mv)" is interpreted as "The i^{th} Case found by **CA** for the main verb mv is $LABEL_i$, which is realized by the substring $string_i$ making up the syntactic group sgr_i".

EXAMPLE:

CA("John broke the window with a hammer") ➵

[$Case_1$: AGT(NP_{subj} : "John", break),
$Case_2$: OBJ(NP_{compl} : "the window", break),
$Case_3$: INST(PP_{compl} : "with a hammer", break)]

Case	**Abbreviation**	**Case**	**Abbreviation**
PARTICIPANT		*SPACE*	
Agent	AGT	Direction	DIR
Beneficiary	BENF	Location at	LAT
Experiencer	EXPR	Location from	LFRM
Instrument	INST	Location to	LTO
Object	OBJ	Location through	LTRU
Recipient	RECP	Orientation	ORNT
CAUSALITY		*QUALITY*	
Cause	CAUS	Accompaniment	ACMP
Effect	EFF	Content	CONT
Opposition	OPP	Exclusion	EXCL
Purpose	PURP	Manner	MANR
		Material	MATR
TIME		Measure	MEAS
Frequency	FREQ	Order	ORD
Time at	TAT		
Time from	TFRM		
Time to	TTO		
Time through	TTRU		

Table I. List of Cases Used in HAIKU.

iii) This notation extends to complex sentences easily. The transformation of a complex sentence into a collection of simple clauses is indicated by '↔', interpreted as equivalence. Parentheses and the symbols ∧ and ∨ denote conjunction and disjunction respectively. A clause whose two main verbs are joined by *and* would be rendered as:

CA(Clause with two "anded" main verbs) ↔
CA(Clause with the first verb) ∧ **CA**(Clause with the second verb).

EXAMPLE ("washed" is analyzed as transitive):

CA("John washed and dried the dishes") ↔
CA("John washed the dishes") ∧ **CA**("John dried the dishes").

2.3.3 Details of the Framework

The following *semantic interpretation rules* express the principles underlying HAIKU's CA process in simple terms. They are derived from introspection on a variety of texts and influenced by Quirk (1985) and other grammarians. The rules attempt to identify the commonest or most reasonable interpretation of a syntactic construction, deliberately focusing on more complicated examples. These heuristics are not always correct.

Complex NPs and embedded clauses

— NPs with post-modifiers:

Ex.1: "John knows the man *who wrote that letter*" (relative clause)
Ex.2: "John knows the man *with the long hair*" (PP)
Ex.3: "John knows Santa Claus, *our federal representative*" (appositive)

NP post-modifiers do not mark a Case at the level of the sentence's main verb. Their purpose is to specify the individual(s) which fill other Cases. The same is true of modifiers within a noun phrase such as "*John's* car" or "*data* file". The relationship between a noun and its modifiers is handled by HAIKU's Noun-Modifier Relationship Analyzer and is outside the scope of this paper. However, a clause which post-modifies a NP requires its own Case Analysis, whose result is attached to the NP. In the following example, $Case_{21}$ and $Case_{22}$ are Cases within $Case_2$. Curly brackets denote a Case structure with embedded substructures.

EXAMPLE:

CA("John knows the man who wrote that letter") ➵
[$Case_1$: AGT(NP_{subj} : "John", know),

{ $Case_2$: OBJ(NP_{compl} : “the man who wrote that letter”, know),
[$Case_{21}$: AGT(NP_{subj} : “the man”, write),
$Case_{22}$: OBJ(NP_{compl} : “that letter”, write)] }]

— Nominal clauses:

Ex.1: “John wants *to go to New York*” (to-infinitive clause)
Ex.2: “John finished *writing his last letter last night*” (-ing clause)
Ex.3: “John knows *that she has copied his homework*” (that clause)

A nominal clause marks a Case because it is a complement of the sentence’s main verb. It plays the same role in the sentence as a NP but is syntactically and semantically more complicated, often inheriting referents from the parent clause. Thus it must be inferred that John is the syntactic subject of *go* in Ex.1 and of *writing* in Ex.2. Nominal clauses require separate Case analysis as well.

EXAMPLE:

CA(“John knows that she has copied his homework”) ➵
[$Case_1$: AGT(NP_{subj} : “John”, know),
{ $Case_2$: OBJ(NC_{compl} : “that she has copied his homework”, know),
[$Case_{21}$: AGT(NP_{subj} : “she”, copy),
$Case_{22}$: OBJ(NP_{compl} : “his homework”, copy)] }]

— Comparative clauses:

Ex.1: “Bob behaves *more intelligently than your sister*”
Ex.2: “Bob works *faster than his colleagues*”

Comparatives mark a Case, usually MANR (see Table I).

EXAMPLE:

CA(“Bob works faster than his colleagues”) ➵
[$Case_1$: AGT(NP_{subj} : “Bob”, work),
$Case_2$: MANR(COMPAR_CL : “faster than his colleagues”, work)]

Stative verbs

Ex.1: “A data file *is* a computer science concept”
Ex.2: “This file *seems* empty”

Most sentences with a stative main verb can be seen as assertions of varying modality: "This file is empty"; "This file seems empty". Such assertions introduce new entries into the hierarchy of concepts used in the current discourse or associate new properties with known concepts. They may also have existential or temporal implications. Clauses with stative verbs do not have Cases; because the complement is really a modifier of the subject NP, such sentences will be handled by the NMR Analyzer.

Structures other than simple declarative clauses

Ex.1: "Here is the solution to Bob's problem" (Here+*be*)
Ex.2: "There is more than one solution to this problem" (existential-There)
Ex.3: "See the solution in the preceding chapter" (imperative)

In Here+*be* or existential-There sentences the main verb is usually a form of *be*. For this reason they are treated like other stative sentences (see 2.2.3.2), which do not mark Cases. An imperative statement is a declarative sentence without a subject[5]. Often it is directed at the reader, the implicit Agent, and contains no new information about the domain of discourse. Only certain imperatives mark Cases, in particular ones indicating the sequence of steps to perform a task. Imperatives require more study to determine what kind of processing they deserve.

Conjoined structures

— Conjoined main verbs:

Ex.1: "Bob *washed and dried* the dishes"

A major problem here is establishing if any order is implied by the construction. *Temporal order* can be implied by verb conjunctions. Consider "Bob *washed and dried* the dishes" and "Bob *dried and washed* the dishes". We usually understand the former as "Bob washed the dishes *and then* dried them" while the dried-and-washed example sounds slightly odd. On the other hand, the actions in "Bob washed the dishes and sang" would probably be interpreted as simultaneous. Conjoined verbs can also convey a *causal relationship*, as in "Bob twisted and broke his ankle" meaning "Bob twisted his ankle, causing him to break it".

---

5 Imperatives in technical texts share several properties with purpose clauses in the context of instruction understanding, see Di Eugenio (1992).

At present HAIKU ignores implicit temporal and causal relationships and reproduces the structure twice with single verbs. Subsequent processes have access to HAIKU's outputs and the original sentence from which to construct temporal or causal models.

EXAMPLE:

CA("Bob washed and dried the dishes") ↔
CA("Bob washed the dishes") ∧ **CA**("Bob dried the dishes")

— Conjoined verb phrases:

Ex.1: "Bob *wrote the letter and sent it to his friend*"

A sentence with conjoined VPs can be split into two or more clauses in which the associated constituents are explicitly represented. CLR Analysis is applied to the resulting conjoined structure and its individual clauses are then Case analyzed. The caveat about implicit order applies here as it does to conjoined main verbs—"Bob's letter" can only be sent after "he wrote it".

EXAMPLE:

CA("Bob wrote the letter and sent it to his friend") ↔
CA("Bob wrote the letter") connector/intra_sentential:and **CA**("Bob sent it to his friend")

— Conjoined noun phrases and nominal clauses:

Ex.1: "Bob broke *the window and the drinking glass*" (conjoined NPs)
Ex.2: "Bob wants *to write a letter and to send it today*" (conjoined nominal clauses)

These conjoined elements are analyzed as a single composite object which fills one Case.[6]

— Conjoined prepositional phrases:

Ex.1: "Bob sent his file *to the A printer and to the B printer*" (the same preposition)
Ex.2: "Bob sent his file *in a prepaid mailer and with an urgent stamp*" (different prepositions)

Only one Case is marked by identical prepositions. Different Cases should be attributed to each phrase headed by a different preposition.

Note: *conjoined clauses* such as "*Although* I admire her reasoning, I reject her conclusions." or "I admire her reasoning *but* I reject her conclusions." are recognized by CLR Analysis.

---

6 Such composite objects can be correctly represented in the Conceptual Network formalism.

3 Interactive Case Analysis for Knowledge Acquisition: The Core of HAIKU

This section gives details of Case Analysis, an important step in TANKA's acquisition of knowledge from English technical texts. Given the parse tree of a clause produced by DIPETT, HAIKU semi-automatically extracts the pattern of Cases that best characterizes meaning of the clause. Extraction is based on the matching of Case markers.

3.1 Data Structures

HAIKU stores patterns of Cases and their markers in dictionaries and refers to them when processing new inputs. An input having few or no markers in common with the patterns in the dictionary introduces a new pattern. If an input matches, HAIKU asks the user to confirm the previous semantic interpretation expressed as a pattern of Cases. The user's response may create a new Case pattern. In this way HAIKU pursues TANKA's goal of acquiring knowledge from text—the knowledge contained in Case patterns is an essential element of the network fragments that add up to compose a conceptual model of the text's domain. Two structures figure in this process:

— **Case-marker Pattern (CMP)**: an ordered list of Case markers linked by hyphens. The resulting expression represents the Case markers appearing in a clause. In our notation, the arrow ➟ associates a clause with a CMP. In HAIKU, a clause can have only one CMP, that is, only one syntactic analysis from which a unique CMP is derived. The symbols `psubj`, `pobj`, `piobj` denote the positional subject, object and indirect object.

  EXAMPLE: "the program ran at midnight" ➟ `psubj-at`
  (*the program* is the subject, the prepositional phrase *at midnight* is the complement).

— **Case Pattern (CP)**: an ordered list of Cases, linked by hyphens. The resulting expression represents the Cases occurring in a clause. The arrow ➨ associates a clause with a CP. In HAIKU, a clause is assigned a single CP. Although in theory a semantically ambiguous clause would have more than one, in practice people settle on a single interpretation as they advance through a text. HAIKU might help the user make such a commitment by showing the surrounding context on request.

  EXAMPLE: "the program ran at midnight" ➨ AGT-TAT
  (*the program* denotes the agent, *at midnight* denotes the time of the activity).

HAIKU stores these structures in a Meaning Dictionary (MDict) and a Case-marker Pattern Dictionary (CMPDict) respectively.

— **Meaning Dictionary (MDict)**: a dictionary of entries for individual words, verbs, prepositions and adverbs. A verb entry contains a list of CMPs found with the verb and lists of the Cases associated with the set of markers in these CMPs. The arrow → associates a CM with the list of Cases it has realized in processing to date. Entries for preposition and adverb Case markers list all the Cases the marker can realize. The lists for these closed category parts of speech are fixed; see details in Barker *et al.* (1993). Here is a sample verb entry:

EXAMPLE:
```
mDict(run,  [ psubj-at, psubj-pobj-at ],
            [  at → [LAT, TAT],
               pobj → [OBJ],
               psubj → [AGT] ] ).
```

— **Case-marker Pattern Dictionary (CMPDict)**: a dictionary of entries indexed on CMP. An entry contains a list of CPs found associated with the CMP in the text. Each CP is illustrated by an example sentence. The arrow ⇨ associates a CP with a clause. In HAIKU a CP may be associated with more than one clause since the same meaning can be realized by more than one syntactic form.

EXAMPLE:
```
cmpDict( psubj-at,
         [  AGT-LAT ⇨ "The program ran at the office"
            AGT-TAT ⇨ "Updates run at midnight" ] ).
```

Although the MDict may be initialized with entries for a number of common CMPs, both dictionaries can also be empty when processing of a new text begins. HAIKU does not need seed knowledge. The cost of starting with empty dictionaries is a good deal of initial interaction with the system to teach it basic knowledge about the domain. As this knowledge is acquired HAIKU becomes perceptibly more adept at making suggestions—both of its dictionaries are updated continuously as text is processed.

Processing associates a unique CP with each clause parsed. To accomplish this, HAIKU first searches its dictionaries for the *target CP* that matches the CP of the input sentence exactly or most closely. Order does not matter in patterns: `psubj-pobj-at-by` is equivalent to `psubj-pobj-by-at` (and for that matter AGT-OBJ-LAT-TAT is equivalent to AGT-TAT-OBJ-LAT). An example sentence to illustrate this CP is then fetched from the dictionary and shown to the user. If the CP and example sentence are not acceptable, the system shifts to a mode in which the user defines a new CP thereby adding to its knowledge (see 3.4).

3.2 An Example

Let's walk through the processing of a text fragment before we turn to the algorithms at work in HAIKU. Consider the following short text about data center operations. Not particularly meaningful or coherent, it has been chosen to demonstrate different aspects of HAIKU's processing:

> "The daily update executes in the morning; this program knows Word Perfect file format. We print hourly, while the system prints all reports at noon. You can run jobs at any time.
>
> The operating system executes print jobs at this site; it also runs a report generator at each local site by datalink. Transaction processing and user programs run at all locations. Although the various operating systems run programs, each process owns sufficient resources."

HAIKU breaks the two paragraphs into ten simple clauses before performing CA. At this stage of processing text cohesion on the discourse level is ignored. It remains latent, however, in the order of clauses output by HAIKU. Although the ten clauses are processed in order of occurrence, they are sorted on main verb in the list below to make it easier to find them in the dictionaries that follow.

the daily update **executes** in the morning
the operating system **executes** print jobs at this site
this program **knows** Word Perfect file format
each process **owns** sufficient resources
we **print** hourly
the system **prints** all reports at noon
you can **run** jobs at any time
transaction processing and user programs **run** at all locations
the various operating systems **run** programs
it also **runs** a report generator at each local site by datalink

If processing began with HAIKU's two dictionaries empty, they will have the following contents when it ends:

```
mDict( execute,  [ psubj-in, psubj-pobj-at ],
     [ at → [LAT], in → [TAT], pobj → [OBJ], psubj → [AGT] ] ).
mDict( know,  [ psubj-pobj ],
     [ pobj → [OBJ], psubj → [AGT] ] ).
mDict( own,   [ psubj-pobj ],
     [ pobj → [OBJ], psubj → [AGT] ] ).
mDict( print,    [ psubj-adv, psubj-pobj-at ],
     [ adv → [FREQ], at → [TAT], pobj → [OBJ], psubj → [AGT] ] ).
mDict( run,   [ psubj-at, psubj-pobj, psubj-pobj-at, psubj-pobj-at-by ],
     [ at → [LAT, TAT], by → [INST], pobj → [OBJ], psubj → [AGT] ] ).

cmpDict( psubj-adv,
        [ AGT-FREQ ⇨ "we print hourly" ] ).
cmpDict( psubj-at,
        [ AGT-LAT ⇨ "transaction processing and user programs run
                     at all locations" ] ).
cmpDict( psubj-in,
        [ AGT-TAT ⇨ "the daily update executes in the morning" ] ).
cmpDict( psubj-pobj,
        [ AGT-OBJ ⇨ "this program knows Word Perfect file format" ] ).
cmpDict( psubj-pobj-at,
        [ AGT-OBJ-LAT ⇨ "the system executes print jobs at this site",
          AGT-OBJ-TAT ⇨ "the system prints all reports at noon" ] ).
cmpDict( psubj-pobj-at-by,
        [ AGT-OBJ-LAT-INST ⇨ "the system runs a report generator at
                              each local site by datalink" ] ).
```

Table II. A Result of HAIKU Processing.

The meaning dictionary has five verb entries and the Case-marker pattern dictionary has seven. MDict entries include the root form of the indexed verb (`execute`), a list of all CMPs it appears with (`[psubj-in, psubj-pobj-at]`), and a second list of all Cases realized by each marker in these patterns (`[at → [LAT], in → [TAT], pobj → [OBJ], psubj → [AGT]]`). CMPDict entries are simpler. They begin with the pattern (`psubj-pobj-at`) and contain a list of all of its CP realizations (`[AGT-OBJ-LAT ⇨ … AGT-OBJ-TAT ⇨ …]`) with an example sentence

for each ([… "the system executes print jobs at this site" … "the system prints all reports at noon"]). Not all input sentences appear as examples, and duplicate CMPs or CPs are eliminated.

3.3 The Process of Case Analysis and Its Algorithm

Let's now turn to the process that produces these outputs and fill in the details of the framework outlined in Section 2.3. If we view that outline as the "what and why" of Case Analysis with learning, the following represents the "how", that is, the CA algorithm. It has been implemented and constitutes the central part of HAIKU.

3.3.1 Converting Multiple-Clause Input to a Series of Clauses

HAIKU's Case Analysis module operates on single clauses. Complex sentences are first decomposed into a list of simple clauses by a tree-transforming algorithm, the semantic analysis driver, which systematically decomposes an arbitrary English input, sentence or fragment, into these simple clauses and the connectives (coordinators, correlatives, subordinators etc.) that join them. The CLR Analyzer assigns semantic labels to the relationships between these clauses based on the particular connective and syntactic features of the conjoined clauses (Barker 1994). Case Analysis is subsequently applied to each individual clause in the input. When parsing fails and fragments are produced, each fragment containing a verb is also considered for Case Analysis. The following paragraph describes the driver's main algorithm.

> SA_driver:
>
> Extract all clauses from the input parse tree and collect them in a list.
>
> Extract their logical organization if there are any clausal connectives.
>
> Individual clauses are recognizable by these syntactic substructures in the input produced by DIPETT: relative clauses, to-infinitive clauses, regular and genitive 'ing' clauses, 'wh-interro-declarative' clauses such as whether/if clauses, and statements[7].
>
> Invoke CLR Analysis to account for semantic relationships between clauses.
>
> Invoke Case Analysis iteratively for each clause.

---

[7] A statement is a regular clause with a subject, main verb and complement. This structure is also used in the definition of other structures that do not explicitly appear here, such as the that-clause (the report clause).

3.3.2 Case Analysis of a Clause

Case Analysis can now be applied to individual clauses with one main verb. The algorithm searches the CMP dictionary (which grows as new patterns are approved and added) using a heuristic based on similarity of form between the input and stored CMPs. If a perfect match is not found, a procedure attempts to find the best partial match of constituents. The technique presented in Delisle *et al.* (1993) is briefly summarized at the end of this subsection.

Case Analysis begins by extracting of the subject, main verb, complement and adverbial modifiers from the clause in a decomposition process akin to that performed by CLRA on the inter-clausal level. Once these constituents have been identified, HAIKU can easily construct the *input* CMP associated with the clause being processed and begin Case Analysis in earnest.

Discussion of the algorithm in Figure 2 uses these terms: the *input verb* and *input CMP* refer to the verb and CMP of the clause being processed; a *new* pattern has not yet been encountered with the input verb, while an *entirely new* pattern has not yet been encountered with any verb.

Depending on whether HAIKU has encountered them before, five meaningful combinations of input verb, CMP and CP are possible. The algorithm labels these situations $\mathcal{S}_i$, i = 1, ..., 5. In each situation HAIKU finds one or more target CPs and identifies suitable example sentences to show the user. Section 3.4 discusses the five situations one by one, illustrating them with references to the extended example presented in Section 3.2.

Sentences do have identifiable meanings, and each parsable clause should associate with a unique CP. To make this connection HAIKU first searches its dictionaries for the CP that most closely matches the input CP. A procedure `find_close` is invoked when the input CMP does not exactly match any previously processed pattern of Case Markers (situations $\mathcal{S}_3$ and $\mathcal{S}_5$ in the CA algorithm). The procedure identifies CMPs similar to the input CMP. When more than one is found, HAIKU must rank them on their closeness to the input CMP. Computation of the closeness of two patterns is based on notions of *undermatching* and *overmatching* among CMPs taken as sets. Intuitively, undermatching occurs when a CMP is a non-empty subset of another CMP: `psubj-pobj` is a non-empty subset of `psubj-pobj-at`. When the best candidate CMP has been identified, HAIKU fetches an example sentence from its dictionary and engages the user in a dialog about the syntax and semantics of the input sentence. Its questions are based on those linguistic elements HAIKU has previously encountered in the text which are most similar to elements in the new sentence. At the end of the dialog the user agrees that the CMP and CP the system proposes are correct.

```
IF input verb is in MDict                          -- known verb V
THEN
    IF input CMP is among CMPs in MDict entry for input verb
                                                   -- known CMP already associated with V
    THEN                                           -- known CP or new CP with known CMP
S1:     dialogue about CPs in CMPDict entry for input CMP;
        update dictionaries (1);
    ELSIF input CMP is in CMPDict                  -- known CMP not yet associated with V
    THEN                                           -- new or known CP
S2:     dialogue about CPs in CMPDict entry for input CMP;
        update dictionaries (2);
    ELSE                                           -- entirely new CMP
        find_close(input CMP, close CMPs);
        ask user to select closest CMP from among close CMPs;
S3:     dialogue about CPs in CMPDict entry for closest CMP;
                                                   -- new or known CP
        update dictionaries (3);
    END IF;
ELSE                                               -- new verb
    IF input CMP is in CMPDict                     -- known CMP
    THEN                                           -- new CP
S4:     dialogue about CPs in CMPDict entry for input CMP;
        update dictionaries (4);
    ELSE                                           -- entirely new CMP
        find_close(input CMP, close CMPs);
        ask user to select closest CMP from among close CMPs;
S5:     dialogue about CPs in CMPDict entry for closest CMP;
                                                   -- new CP
        update dictionaries (5);
    END IF;
END IF.
```

Figure 2. The CA Algorithm.

3.4 Five Processing Scenarios

3.4.1 Dealing with New CPs

The *target* CP (Section 3.1) is first sought in the CMPDict. CPs and example sentences illustrating them are presented to the user until he selects a sentence whose CP he deems to most closely match the meaning of the current input. If none is acceptable, HAIKU works with the user to build the CP from individual CM → Case mappings. This is summarized in the following procedure.

> `new_CP_dialogue:`
>
> None of the proposed CPs has been accepted by the user. Interaction can proceed down two alternative avenues.
>
> Search the MDict entry for each Case marker. Show the user a list of Cases covered by this marker. When he selects one, add the selection to the new CP. When all Case markers have been linked to Cases, store the new CP in the input CMP's entry. Use the current sentence as its example.
>
> OR use `find_close` to identify the CMP closest to the input CMP. Continue as if this new CMP were the input.

Finding an example sentence is easy if the target CP is identical to a stored CP: HAIKU uses the sentence that was stored as a result of previous processing. Otherwise, the example sentence could be derived by adding or deleting constituents in the sentence stored with the chosen CP. Derived sentences, however, may not be well-formed, so at present only actual input sentences are stored with new CPs.

3.4.2 $\mathcal{S}_1$: known verb, known CMP, known or new CP

Fetch CPs from the CMPDict entry for the input CMP[8]. Present the CPs and their example sentences to the user. Usually one is selected; otherwise call `new_CP_dialogue`: we have encountered a new CP.

— **analyze Cases (1)**

---

[8] The presentation in Sections 3.4.2 to 3.4.6 uses a style which is more imperative (that is, command-like) than elsewhere: this is suitable for the CA-algorithm description at this point in Section 3.

Assume the input sentence is `User programs can print reports at high speed`. The input verb is `print` and the input CMP is `psubj-pobj-at`. Assume that a search of MDict finds an entry for `print`, and within that entry, a match for the input CMP. This success sends HAIKU to the CMPDict where it retrieves the CPs associated with `psubj-pobj-at`. There are two: `AGT-OBJ-LAT` and `AGT-OBJ-TAT`. Although both candidates' `AGT` and `OBJ` match the meaning of the input CMP, `at high speed` expresses the `MEAS` Case (Measure), identifying a new CP.

— **update dictionaries (1)**

Because both input verb and input CMP are in HAIKU's dictionaries, there is a possibility the input Case Pattern may also already be known. If so, updating would only occur if the user chooses to replace the current example sentence with the input sentence.

MDict: any Case not already associated with an input Case marker is added to the marker's list. In our example, this means that `MEAS` is added to the list for `at`, producing `at` → `[MEAS, LAT, TAT]`.

CMPDict: a new CPs and the input sentence are added to the list associated with the CMP. For our example `AGT-OBJ-MEAS` ⇨ `"User programs can print reports at high speed"` is added to the entry for `psubj-pobj-at`.

3.4.3 S_2: known verb, new CMP, new or known CP

The CMP has not been encountered with the input verb but has been found with other verbs. A list of candidate target CPs is retrieved from the CMPDict and proposed to the user. If none is accepted, initiate `new_CP_dialogue`.

— **analyze Cases (2)**

Assume the input sentence is `Programs run quickly`. The input verb is `run` and the input CMP is `psubj-adv`. A search of MDict finds no association for this pattern with `run`, except the following CMPDict entry: `[ AGT-FREQ ⇨ "We print hourly" ]`. This single item is shown to the user, who rejects the assignment of `FREQ` and instead links `quickly` with `MANR`.

— **update dictionaries (2)**

MDict: add the new CMP to the input verb's list. If a new CP has been found, add new Cases to the lists of existing Case markers and associations for any new Case marker → Case mappings. In terms of our example that means adding `psubj-adv` to `run`'s entry, producing

`[psubj-pobj, psubj-at, psubj-adv, psubj-pobj-at, psubj-pobj-at-by]`. The example also adds a single mapping, `adv` → `[MANR]`.

CMPDict: the CMP is known, so it already has an entry. If CA processing has produced a new CP, add it and the input sentence to the entry's list of CPs.

3.4.4 $\mathcal{S}_3$: known verb, entirely new CMP, new or known CP

The input CMP has never been encountered before. One or more neighbouring candidates must first be found using the closeness metric (Section 3.3.2). When the best candidate has been found, HAIKU proceeds with processing as in situation $\mathcal{S}_2$.

— **analyze Cases (3)**

Assume the input sentence is `The typical session runs by dialup`. The input verb is `run` and the input CMP is `psubj-by`. A search of MDict finds that this pattern has not been associated with this verb before; searching of CMPDict shows that it has not been associated with any other verb. Lacking any exactly matching CPs to show the user, HAIKU must search for likely candidates in the CMPDict. It calls `find_close` to do this. The list of CMPs which this procedure returns is headed by `psubj-pobj-at-by`. The single CP in this entry is shown to the user, who accepts its mapping of `by` → `INST` as correct.

— **update dictionaries (3)**

MDict: add the new CMP to the input verb's list. If a new CP has been found, add new Cases to the lists of existing Case markers and associations for any new Case marker → Case mappings. Because the three associations in our example already appear in `run`'s entry, no updating is necessary.

CMPDict: add an entry indexed on the new CMP. Use the input sentence as the example for its single CP.

3.4.5 $\mathcal{S}_4$: new verb, known CMP, new CP

The input contains a new verb with a known CMP. HAIKU looks for candidate CPs in the CMPDict and suggests these to the user. A new CP has been encountered, so `new_CP_dialogue` is called.

— **analyze Cases (4)**

Assume the input sentence is `Sessions process concurrently`. The input verb is `process` and the input CMP is `psubj-adv`. Searching MDict establishes that HAIKU has never encountered this verb before; however, a query to CMPDict retrieves the input CMP. This provides a likely CP to show to the user: `AGT-FREQ`. The user edits the candidate to replace `FREQ` with `MANR`.

— **update dictionaries (4)**

MDict: add a new entry to this dictionary containing the input CMP and the approved CP. For `Sessions process concurrently` this means adding

`process, [ psubj-adv ], [ adv → [MANR], psubj → [AGT] ].`

CMPDict: an entry always exists for the input CMP in this situation. If the approved CP is new, add it and use the input sentence as its example. If the CP is already associated with the CMP, give the user a choice between the existing example and the input sentence.

3.4.6 S_5: new verb, entirely new CMP, new CP

The input contains a new verb with an entirely new CMP. One or more neighbouring candidates must first be found using the closeness metric. HAIKU suggests the associated CP associated with the closest CMP and `new_CP_dialogue` is called because a new CP has been encountered.

— **analyze Cases (5)**

Assume the new sentence is `Many users require backups for audits`. The input verb is `require` and the input CMP is `psubj-pobj-for`. Neither item is found in its dictionary, so HAIKU must search for the CMP closest to `psubj-pobj-for`. `find_close` returns the not-very-helpful `psubj-pobj`. Lacking any recommended association for the preposition `for`, the user might well choose to pick a Case from the list of all possible candidates, selecting `PURP` as the Case marked by `for`.

— **update dictionaries (5)**

MDict: add a new entry for the input verb; add the new CMP to its list. Add associations for all markers and their (single) Cases. For `require`, `psubj-pobj-for` is the CMP and the three associations are `[ for → [PURP], pobj → [OBJ], psubj → [AGT] ]`.

CMPDict: add a new entry for the input CMP. Store the input sentence as the approved CP's example sentence. In our example this means adding `psubj-pobj-for, [AGT-OBJ-PURP` ➪

"Many users require backups for audits"]. If the CP is new, add the verb to the list of verbs associated with the CP and select one example as in situation S_1.

3.5 Difficult Issues

Two aspects of the CA process pose sufficient problems to warrant special mention. These are the task of finding example sentences which illustrate the current instance accurately, and the inherent complexity of real world semantics.

3.5.1 Finding and Using Example Sentences

Avoiding complete rejection

Case Analysis requires user cooperation. At present the user can only reject an inappropriate example sentence. The system learns nothing from a total rejection, however, so we envision letting the user identify faulty constituents in the example and build on correct ones. This will ease the task of identifying suitable example sentences and correct CPs.

Consider "They can execute the program at will". Suppose `execute` is a new verb but `psubj-pobj-at` has already been encountered (situation S_4). Further suppose the CMPDict shows that `psubj-pobj-at` may correspond to either AGT-OBJ-TAT or AGT-OBJ-LAT, and that both these patterns previously occurred with `run`. The CMPDict provides the example sentences "The system prints all reports at noon" and "The system executes print jobs at this site" for the two alternative readings.

HAIKU would present the sentences to the user. Both would be rejected because neither includes the Case expressed by `at will`. With no more stored examples to propose HAIKU might try to generate new ones, but this would involve the difficult task of reliably generating meaningful sentences. A future extension of HAIKU might instead ask the user to identify incorrect elements in the current example sentence (see the example at the end of this subsection). Once the user has identified `at will` as the culprit element and endorsed the mapping of `psubj` to AGT and `pobj` to OBJ, HAIKU would then suggest alternative Cases that `at will` can realize, among them the correct MANR. The user's recognition of this would allow the system to learn that "They can execute the program at will" is an example of the newly-learned mapping, `psubj-pobj-at` to AGT-OBJ-MANR.

Problems with the closeness metric

Patterns produced using the closeness metric in situations $\mathcal{S}_3$ and $\mathcal{S}_5$ pose additional problems for the production of apt example sentences. Suppose the example sentences for AGT-OBJ and AGT-OBJ-LAT-MANR are unsatisfactory and HAIKU must base another on "Bob can run the software at home", the example stored for AGT-OBJ-LAT. To use this example for AGT-OBJ we need to discard LAT and its corresponding marker, converting the sentence into "Bob can run the software". Conversely, to use it for AGT-OBJ-LAT-MANR we must add MANR and a corresponding Case marker, producing "Bob can run the software at home *with ease*".

The question is whether adding or removing Case markers and their fillers produces reasonably well-formed and meaningful example sentences. Removing Case realizations appears less troublesome than adding them: in fact, the closeness metric makes just this assumption. In any event the operations described in the previous paragraph seem straightforward, given a record of the mappings between the elements in CMPs and CPs for each example sentence. It remains to be seen whether ease of production will be matched by usefulness.

A possible scenario

HAIKU's dialogues are designed to extract a maximum of information with a minimum of effort[9]—a single interaction can identify more than one error in the semantic analogy between the input and the example sentences. As Figure 3 illustrates, a user can approve the analogy (option 0), adjust the suggested semantic pattern (option 1), look at any other example sentences (option 2), or even build a new CP from scratch (option 3).

3.5.2 Challenging HAIKU: a Difficult Example

One of the original questions TANKA sought to investigate is the strength of association between individual verbs and CPs. Is Location_at more likely to occur with *run* than with *think* ? This question has had a direct impact on the design of HAIKU. Here is an example.

---

9 See Brown & Nirenburg (1990) for an interactive semantic disambiguation module called an augmentor that deals with multiple parses.

Does the sentence (current input)

"The man put the parcel in the car yesterday"

resemble (example sentence)

"Every day, dogs chase frisbees in the park" ?

where

a)	the man	->	dogs	(agent)
b)	the parcel	->	frisbees	(object)
c)	in the car	->	in the park	(location at)
d)	yesterday	->	every day	(frequency)

0) OK.
1) Adjust this pattern ...
2) 3 other patterns exist. Look at an example sentence for one of them.
3) Build a completely new pattern.

Enter a choice: _

Figure 3. A Hypothetical Interaction with the User During Semantic Analysis.

Suppose *take* is the input verb and `psubj-pobj-from-to` the input CMP. Assume the CMP dictionary has `psubj-pobj-to` associated with *take* and maps AGT-OBJ-BENF to it. Assume further that the dictionary also associates `psubj-pobj-from-to` with *throw* and AGT-OBJ-LFRM-LTO. What association should be chosen for the input preposition *from* ? The BENF in the closely-matching pattern found with the same verb, or the LFRM in the exactly-matching pattern found with a different verb? This is the situation:

INPUT	take:	`psubj-pobj-from-to`	
DICTIONARIES	take:	`psubj-pobj-to`	AGT-OBJ-BENF
	throw:	`psubj-pobj-from-to`	AGT-OBJ-LFRM-LTO

The current version of the CA algorithm would reason as follows:

a) *take* is a known verb—this is one of the situations $\mathcal{S}_1$, $\mathcal{S}_2$, or $\mathcal{S}_3$;

b) it is not situation $\mathcal{S}_1$ because `psubj-pobj-from-to` has never occurred with *take* before;

c) it is situation $\mathcal{S}_2$ because `psubj-pobj-from-to` is a known pattern not associated with *take* .

The algorithm would thus suggest AGT-OBJ-LFRM-LTO as the target CP.

Would a hybrid pattern like AGT-OBJ-LFRM-BENF be superior? Or should HAIKU begin with AGT-OBJ-BENF and negotiate with the user on which Case to map to *from*? A hybrid is not guaranteed to be semantically consistent, but using …-LFRM-LTO rather than …-BENF tends to misdirect the user. It can be argued that HAIKU should try to find a close pattern with the same verb before switching to a different one. This would result in the target CP being AGT-OBJ-BENF.

We believe HAIKU should use the closeness metric only after trying all possible perfect matchings with the same verb, and then with different ones. This minimizes computation and the difficult task of generating good example sentences for hybrid CPs. In summary, the CA algorithm currently gives highest priority to CMPs already encountered with the input verb, employing the CPs of closely-matching CMPs associated with other verbs only if necessary.

4 Related Research: Acquiring Knowledge Through Text Processing

In this section we review other work on the acquisition of semantic and domain-related knowledge which is relevant to this paper. The discussion is organized thematically and is limited to research that deals with processing real-world texts. For instance, Kaplan (1989) and Dick (1992) both assume that the text has already been encoded or parsed manually and proceed directly to semantics. We believe this is too much to presume in a practical system.

4.1 Acquisition by learning

Mooney's GENESIS system (1990) shares similarities with FRUMP (DeJong 1982). Although it also uses an approach derived from Machine Learning (ML), it is aimed at understanding and explaining short stories and is not suited to expository[10] texts. According to Mooney (1990: 96): "Explanation-Based Learning in GENESIS can therefore be viewed as the acquisition of schemata that allow the system to process narratives using efficient schema-based techniques which previously could only have been understood using inefficient, search-intensive, plan-based understanding.". GENESIS depends on *a priori* knowledge; understanding is

---

[10] "… discourse intended to give information about or an explanation of difficult material" American Heritage Dictionary, 3rd edition.

impossible if a narrative contains actions that do not suggest already known schemata. In ML terms, this is deductive learning (Dietterich 1989).

Zelle and Mooney (1993) and Aliprandi (1993) show how different learning techniques, Inductive Logic Programming and Standard Induction, can be applied to the problem of prepositional phrase attachment. A separate research community (Powers & Turk 1989) focuses on the difficult question of using machine learning to understand the cognitive aspects of language acquisition. Cohen (1990) shows how crucial a flexible definition of operationality is to learning from texts. His work concentrates on learning without attempting to develop an integrated NLP-ML system.

4.2 Classification tasks

Classification is an essential first step in model construction—things must be differentiated before complex models can be built on their relationships. Several methods have been proposed for classification tasks in KB construction. They deal with the extraction of more or less complex classification knowledge from different types of textual material.

The systems of Gomez (1989) and Reimer (1990) produce conceptual structures. Gomez' system has a relatively powerful linguistic processor which seems biased towards clauses whose main verb is 'be' (generally expressing *is-a* relations). Gomez (1989: 15-12) makes a remark about *a priori* knowledge that echoes our view: "If the KA system is going to be domain-independent, then the system cannot have *a priori* knowledge about concepts which are domain specific". His approach lets users define missing domain-specific concepts by relating them to existing ones via *is-a* and *part-of* relations. Reimer's system uses a very simple parser and inductively generalizes concepts learned from text to create more generic ones. It does this with the help of domain-dependent background knowledge and the semantics implicit in its representation model.

4.3 Special forms of text

A different approach is apparent in the work of Rinaldo (1989) and Moulin & Rousseau (1992). Their systems reduce text processing to its most basic form, simply scanning input sentences for predetermined fixed patterns such as 'if', 'because', 'when' and decomposing the sentences around these keywords into KB elements or representations that stand for production

rules[11]. Output is still in natural language in Rinaldo's system, while Moulin and Rousseau produce a structured derivative form. Their representations are meant to be used by rule-based systems after further processing. Both systems can only learn rules which are expressed in the surface form of the input text.

Rinaldo's system is not interactive and requires significant *a priori* knowledge—all words must be known and assigned a conceptual category beforehand. This precoded knowledge is also used to determine whether a given input sentence is meaningful (that is, whether the system should try to extract a rule from it). The system was intended to test the feasibility of automatically generating rules from a medical text. The user is expected to verify whether the rules produced are correct and relevant. The work of Gao & Salveter (1991) is similar: they propose an Automated Knowledge Engineer to acquire knowledge for taxonomic expert systems.

Moulin and Rousseau's work differs in being highly interactive and in depending heavily on the participation of the user, who performs all interpretation. Their system is more a support tool than a comprehensive NLP system. It helps a human user derive deontic rules from prescriptive texts by "semi-intelligently" carving up the text without attempting to represent (understand) what it manipulates. Accordingly, this system does not perform detailed parsing or semantic analysis**,** and can only process highly formatted texts such as regulations.

Rousselot *et al.* (1992) take a similar but ostensibly more general approach to extracting knowledge from scientific texts. They also aim at constructing a KB of elements distilled from texts, but without natural language grammars, parsers or analyzers. They simply assume that scientific texts are written with a very rigid and predictable syntax and thus propose to extract knowledge from them using a number of rules based on specific constructs and keywords. This is reminiscent of the technique used more than 25 years ago in ELIZA (Weizenbaum 1967).

4.4 Selectional pattern acquisition, thematic knowledge acquisition, and lexical acquisition

Lang & Hirschman (1988) describe the SPQR (Selectional Pattern Queries and Responses) module of the PUNDIT text processing system. Although SPQR seems aimed at disambiguating

---

[11] This statement is not entirely fair to Rinaldo's system which uses the Linguistics String Parser (see Sager 1981) and performs simple discourse processing. However, there remains no trace whatsoever of this information (especially syntax and semantics) in the output of his system. This is surprising when one considers that to make such rules amenable to knowledge-based processing, one would use elements of such information to transform the system's output into an appropriate knowledge representation.

parses rather than modelling conceptual domains, it has several high-level features in common with TANKA. Its authors consider many of the same problems: the incremental acquisition of domain-specific knowledge from text, the use of a broad-coverage grammar[12], eliciting information from the user, and the ease of porting to other domains.

Despite the similarities Lang & Hirschman's approach differs from ours in several important respects. Their objective of acquiring domain knowledge to rule out semantically anomalous parses requires SPQR to learn what are essentially selectional constraints. Our goal is more ambitious. We hope to acquire generic knowledge substantially broader than selectional constraints. Thus users interact with SPQR about the admissibility of selectional constraints, and with TANKA about the correctness of conceptual meanings extracted from text. Second, the user must classify the selectional constraints SPQR acquires as either permissible or impermissible, good or bad. These alternatives may not always be correct, and it appears impossible to change them once made. Third, SPQR's generalization of selectional constraints into Case frames relies on the advance availability of a KB offering a complete *is-a* model of the domain. This model is static and unaffected by the system's operation or the user's participation. TANKA presumes only the highest levels of the domain model and constantly updates it with the results of processing.

The work of Liu and Soo (1993) deserves special mention in relationship to Case Analysis in HAIKU rather than TANKA's overall goals. The authors propose a method of acquiring domain-independent, thematic knowledge by exploiting syntactic clues in training sentences. Their system's goal is to collect the information needed to discriminate thematic roles in input strings (for example, to propose the argument structure [Theme Goal] for the verb 'go') and eventually to build an extensible thematic dictionary. Syntactic ambiguities are resolved by the trainer or simply avoided by taking input from a syntactically pre-processed corpus. The syntactic properties of the thematic roles—whether a role can be associated with say an animate argument—serve as a preliminary filter to reduce the hypothesis space of possible thematic roles for arguments in training sentences. Other heuristics are used to reduce ambiguities further. Thus the Imperative Heuristic says that "imperatives are permissible only for Agent subjects".

Several elements of Liu and Soo's work resemble features of HAIKU: the goal of acquiring domain-independent knowledge; a reliance on detailed parse trees; use of a fixed list of sentence thematic roles, syntactic clues and other heuristic rules; and associating thematic structures with

---

12 They consider the parsing of telegraphic text elements whereas we do not. This probably explains the emphasis of their approach on parsing disambiguation. However, their approach does not seem to consider the problem of multi-sentence fragments (only single-sentence) whereas we do.

verbs (like our Case Patterns). Their work also differs from ours in several important ways. First, they identify only 13 thematic roles. It is not clear if this set is complete or intended to be universal. Second, Liu and Soo's system does not seem to maintain the equivalent of Case-marker Patterns. This is a serious disadvantage: such patterns would help discriminate in the search for syntactic clues. Third, their system is designed to find a verb's single good thematic structure whereas we consider all different Case Patterns. Because their system cannot determine whether an argument is optional or not—whether it is *adjunct* or *required internal* —the goal of determining only good thematic structure appears questionable and may restrict the utility of this approach. On the other hand, their system can exploit corpora to find candidate argument structures for verbs that will be used to generate sentences whose semantic validity is judged by the trainer. Finally, interaction with the trainer is well-designed and intuitive.

Other approaches deal with automatic lexical acquisition from texts. Cardie (1993) presents a system that learns part-of-speech, sense and concept activation knowledge for all open class words in a corpus—*concept activation* indicates which domain-specific concept if any is activated by a word. During its initial training phase the user guides the system to create a case base of context-sensitive word definitions. Riloff (1993) presents AutoSlog, a system which automatically constructs a domain-specific dictionary of concept nodes (frames with slots) for extracting information from text. This is done by a type of text skimming called *selective concept extraction*. The system requires a part-of-speech lexicon and an initial training period.

4.5 Extraction from corpora and statistically-based methods

Recent years have seen a growth in interest in extracting information from corpora—*Computational Linguistics*, **19**(1-2)—and from dictionaries (Montemagni & Vanderwende 1992, Dolan *et al.* 1993). One particularly active line of research is extracting patterns, be they lexical, syntactic or semantic, from text which has been tagged for part-of-speech. Tagging may be manual or automatic (DeRose 1988) using varying amounts of *a priori* knowledge. Much of this work has a statistical basis, so results only make sense if the number of occurrences of a given pattern is relatively large. These methods pose several questions: how can their reliability be measured? how sensitive are they to random error? how do they behave when applied to increasingly larger texts? what are the criteria for constructing a valid corpus?

A number of pattern-matching systems merit detailed discussion. The argument can be made that a system combining the consistency and methodicalness of automatic processing with human expertise has the potential of outdoing an entirely mechanical system while retaining advantages over one that simply stores an expert's knowledge. Such a hybrid system could

acquire knowledge from the text automatically and exploit the user's expertise to validate suggested interpretations without requiring him to decide what knowledge should be acquired.

Zernik & Jacobs (1990) collect word collocations—words tending to occur together—in particular those including the main verb, to improve parsing accuracy and support the assignment of thematic roles in semantic analysis. Zernik (1992) subsequently applies this approach to the problem of tagging. Smadja (1991) suggests filtering raw collocations (called *n-grams* when they contain more than two words) with robust parsing in order to eliminate invalid candidate n-grams. He uses this technique in an information retrieval context. Smadja & McKeown (1991) show how the technique can be adapted to guide lexical selection in language generation. The idea of combining statistics with NLP to perform knowledge acquisition from raw text is also considered in Jacobs (1992), who notes that "the statistical methods themselves must be an aid rather than a replacement for knowledge acquisition" (Jacobs 1992: 182).

Brent (1991a) describes a program that produces a partial list of the verbs in an untagged text together with the simple subcategorization frames in which they occur. His algorithm has verb detection, subcategorization frame detection, and subcategorization frame decision phases. The last stage uses statistical models of word frequency distributions. A later variant (Brent 1991b) tackles the task of automatically classifying verbs as stative or active.

Basili *et al.* (1992: 96) argue that statistical pattern-matching approaches "are based on the (strong) assumption that syntactic similarity in word patterns implies semantic similarity", an assumption they take issue with. They find that "the major problem with collocations is that reliable results are obtained only for a small subset of high-frequency words on very large corpora, otherwise the association ratio becomes unstable" (Basili *et al.* 1992: 97). The authors are interested in learning selectional restrictions to support the design of computational lexicons. Grishman & Sterling (1992) have similar goals and propose a syntax-first strategy for studying word associations which is based on segmentation parsing and semantic hand-tagging.

4.6 Knowledge extraction from texts

Ciravegna *et al.* (1992) present the SINTESI system. It extracts knowledge from four or five-sentence diagnostic reports that describe car faults in Italian, summarizing their technical content and supporting a KB of faults. Information sought by SINTESI includes the main fault, the chain of causes, the chain of effects and the list of parts involved. The system tries to identify the important physical objects mentioned in a report and to discover the relationships between them. SINTESI's methodology is typical of sublanguage NLP strategies. Although its parser covers a

rather limited amount of Italian syntax, it can cope with ill-formedness in a realistic manner. Parsing is essentially semantics-driven, guided by a search for specific semantic markers stored beforehand in the lexicon. Descriptions of all objects of interest are also provided in advance. The sort of knowledge extraction performed by SINTESI is thus closer to text summarization than to knowledge acquisition as performed in TANKA: there is no incremental KB augmentation.

Hauptmann (1993) tackles the problem of deriving a meaningful semantic frame from its associated syntactic tree. His system can function in two modes. In interpretation mode, it uses its KB and a simple generalization strategy to produce a frame representation from an input parse tree. In learning mode, the system is given an input parse tree and its correct and unambiguous frame representation. It then produces rules which map the relevant relationships between the two, accessing its KB as necessary. For complex natural language inputs, however, the cost of creating the ideal frame representation appears to be significantly higher than the effort required to create the mapping rules directly.

Hauptmann's work is relevant in that it is also syntax-driven; we too are in the business of extracting meaning from structure. Our means and ends are quite different, however. Hauptmann's design counts on the availability of a complete domain-specific KB in which all essential selectional constraints are predefined as restrictions on slot fillers. Knowledge acquisition of the kind we are interested in does not occur. Hauptmann acknowledges that his system was not designed for easy portability to other domains (Hauptmann 1993: 15) and in fact it is not clear how it would be applied to less contrived domains than the doctor-patient one used in the project. Sublanguage situations may be its only appropriate context.

Kim & Moldovan (1993) describe PALKA, a semi-automatic knowledge acquisition system designed to aid construction of a large KB of semantic patterns extracted from corpora. Although PALKA's goals are, broadly speaking, similar to those of TANKA, its implementation is quite different. Kim and Moldovan's system relies exclusively on segmentation to process text. It cannot exploit simple syntactic clues like prepositional phrases marking verb semantic roles. Next, the user must direct the system to likely inputs; PALKA then uses a keyword strategy to verify their relevance. Once an input has been segmented the system 'understands' it by matching its phrasal elements to patterns found in the KB. PALKA also requires much *a priori* knowledge: general and domain-specific concept hierarchies, keywords and frame definitions telling the system what to look for in a text, and a dictionary augmented with links between its entries and concept hierarchies. Finally, the system acquires knowledge one event type at a time, that is, all 'bombing' events must be processed together rather than incrementally as encountered in the text.

5 Evaluation and Future Work

This section presents results of using the implementation as described and casts an eye to the future. Although HA1KU cannot be exercised completely at the time of writing because the Noun-Modifier Relationship Analyzer is not fully operational, a good deal of stand-alone testing has already been done. The most significant trends discovered are reported in Section 5.1—details are presented in Delisle (1994). Section 5.2 lists major items to work on in future.

5.1 Evaluation

HA1KU has been tested extensively in isolation. First, 600 test inputs were processed to establish that HA1KU's Case Analyzer could handle all the different types of output DIPETT can produce with the user's help as appropriate. The module was then tested on many segments of the QUIZ manual[13] and the T4043 Tax Guide[14] to fine-tune the CA process and to measure how much help HA1KU needed to function as a useful tool. Finally, a separate experiment was conducted on the QUIZ manual. It is discussed below. We are currently testing DIPETT and HA1KU on the T1-General Individual Tax Guide for Ontario, a chapter from a book on acoustics, and a scientific radio lecture on weather.

We experimented with HA1KU on the first two chapters of the QUIZ manual. The chapters were first parsed with DIPETT and the parse trees saved to a file. Approximately 92% of the 272 inputs in these two chapters produced a full or fragmentary parse tree. HA1KU was then run on these parse trees. Processing began *with empty dictionaries* —all HA1KU had was its list of prepositions and Cases they realize (Barker *et al.* 1993), information which is independent of the domain. Here are the key results.

— The price for starting with an empty KB was an initial high level of tedious user interaction: because HA1KU had not yet accumulated any knowledge, it relied almost exclusively on user input to perform semantic analysis.

---

13 This manual is a guide to Cognos' PowerHouse QUIZ report writer, a component of a fourth generation package. The 1986 version, of which we had an on-line copy, contains seven chapters, which represent some 135 pages, plus three appendices, a glossary, and an index. We have restricted ourselves to the testing and processing of the first six chapters, that is, the seventh and the three appendices simply contained too much QUIZ code or other non-textual material and, thus, were not appropriate for English parsing. The resulting text contains 10863 words and an average of 14.2 words per sentence.

14 The 1990 Canadian T4043E Tax Guide is a supplementary document on child care expenses, as a complement to the general Tax Guide. It has some 15 pages and contains three chapters. We discarded the second chapter since it contains mostly boxed text presented in a non-linear fashion with many visual pointers. The remaining text contains 5773 words and an average of 18.9 words per sentence.

— The initial learning phase became much less demanding after HAIKU had processed some 50 inputs. At that point user interaction began to tend more towards confirmation of proposed semantic analyses than input of new information.

— Analysis was reasonably fast: it took an experienced user approximately 4 to 5 hours to analyze chapters 1 and 2 with HAIKU. A less experienced user would probably have needed a full day.

— Familiarity with the details of the topic being discussed in the text seems less important than a basic understanding of English. In fact domain knowledge helped less than we had anticipated.

— Even if the parse tree is imperfect or the initial analysis erroneous, the user can enter or select the correct Case Marker and Case patterns and ensure that HAIKU's dictionary knowledge remains correct.

— Certain syntactic patterns—a verb with a subject, direct object, and an indirect object or an adverbial complement or modifier—cover such a wide variety of forms that they cannot be used to discriminate amongst semantic patterns very successfully.

Here are the results in detail:

The Meaning Dictionary. Verbs with the largest number of different CMPs are *access* (5), *ask* (4), *enter* (7), *link* (6), *report* (7), *tell* (5), *use* (6), *want* (7), *work* (6). Two different but complementary groups appear to be present: *central domain-specific* activities such as *access*, *link*, *report*; and general purpose explanatory verbs such as *ask*, *enter*, *tell*, *use*, *want*, *work*. 66 verbs (55%) had a single CMP, 27 (23%) had two, and 17 (14%) had three CMPs.

Case Marker Patterns. The vast majority of the 48 CMPs produced by HAIKU have three markers or less. 9 (19%) have 1, 22 (46%) have 2, and 14 (29%) have 3. Two CMPs have 4 markers and one has 5. The number of CPs associated with a given CMP is also of interest. 36 of the 48 CMPs (75%) have a single associated CP; 10 (21%) have 2; and no CMP has 3 or 4. `psubj-pobj-piobj` and `psubj-pobj-adv` have 5 and 7 different CPs, however. This seems to confirm our common-sense expectation that certain syntactic patterns do not help discriminate semantic patterns very much.

Case Patterns. The simplest CPs are associated with the largest number of verbs: `agt-obj` and `agt-obj-manr`; `obj` and `obj-manr`. The latter two CPs illustrate the importance of the imperative form in a user manual like the QUIZ text. 31 (62%) CPs are associated with a single verb; 7 (14%) with two, and 5 (10%) with three verbs.

These results provide some support for our claim that Case Analysis can usefully extract knowledge from texts. Its patterns seems able to discriminate between most verbs except for the simplest instances. These would benefit greatly from the addition of semantic markers for their fillers such as those found in WordNet (Miller 1990) categories for the nominal groups. This topic is discussed in the following section.

5.2 Future Work

5.2.1 Extending and Augmenting Semantic Analysis

Rules for semantic interpretation (see 2.3.3) should be extended, in particular by determining how to treat imperative statements and questions. Using a solely syntax-driven analysis makes these issues particularly difficult.

HAIKU currently deals with semantic interpretation at two syntactic levels: Clause-Level Relationship Analysis operates at the inter-clausal level, Case Analysis at the intra-clausal level. The third step is to look at relationships inside noun phrases, the building blocks of clauses. The design of Noun-Modifier Relationship Analysis closely mirrors that of the Case Analyzer. It first constructs a list of semantic relationships by exhaustively searching for lexical and syntactic items that mark them. This list is similar to the list of Cases. Next it identifies linguistic clues that figure in algorithms for semantic interpretation. As in Case Analysis and CLR Analysis, the emphasis is on deriving as much information as possible from the text in order to make intelligent suggestions for the semantic roles played by parse structures.

5.2.2 A Parse Tree Editor with Learning Capabilities

The Editor

DIPETT does not always parse correctly. Two common errors are *misattachment* of a parse component, and legitimate but *unlikely* analysis due to lexical ambiguity. Such misparses seriously affect semantic analysis, and we plan to allow the user to edit and correct parse tree errors in the near future by entering simple commands to rearrange elements of the existing structure. Directly typing in new structure is not envisaged. At the end of this stage of processing the parse tree will correctly reflect the user's understanding of the syntax and semantics of the sentence it represents.

For example, DIPETT misparses "This would produce the names of the two Boston employees who joined the company *since the start of 1984*," misattaching the last PP (italics) to

the NP “the names”. It should modify the verb “joined” instead. To help the user fix this, the parse tree editor will show the phrases composing DIPETT’s output and ask whether changes are required. The user can then accept the parse as diagrammed or manipulate it with editing commands. A friendly front-end, perhaps a Graphical User Interface, is a key element in this process. And because the user cannot be expected to read DIPETT’s parse trees, a less detailed and more intuitive representation is needed, grained just finely enough to allow the user to repair errors. Figure 4 presents a hypothetical interaction to correct the example sentence. The user’s input appears in boldface. “X ➥ Y” means “X is modified by Y”.

INPUT: “This would produce the names of the two Boston employees who joined the company since the start of 1984.”

SIMPLIFIED PARSE TREE PRODUCED BY DIPETT:

a) This would produce the names
b) ➥ of the two Boston employees
c) ➥ who joined the company
d) ➥ since the start
e) ➥ of 1984

IS THE ABOVE PARSE TREE CORRECT [n/Y]? **n**

PLEASE ENTER MODIFICATION COMMANDS, ONE PER LINE (‘ok’ to exit):

1) **attach ‘d’ to ‘joined’ in ‘c’**
2) **ok**

HERE IS THE RESULTING UPDATED PARSE TREE:

a) This would produce the names
b) ➥ of the two Boston employees
c) ➥ who joined the company
d) ➥ since the start
e) ➥ of 1984

IS THE ABOVE PARSE TREE CORRECT [n/Y]? **y**

Figure 4. A Hypothetical Interaction with the User During Parse Tree Revision

We hypothesize that most parse trees have few errors, and that they are easy to identify and fix. This seems to be the case for misattachments at least. If this hypothesis does hold, producing a single parse tree and repairing it may prove more efficient than picking the right tree from multiple parses (Brown & Nirenburg 1990; Blanchon 1992).

Certain parsing errors may justify parsing a second time, however. A misparse caused by *lexical ambiguity* is one. Identifying a word as a verb instead of as a noun may result in a parse tree may be so different from what ought to be that repairing it with the editor simply does not make sense. A better solution is to update the lexicon with the noun reading and reparse the sentence anew. The editor will therefore include commands to reparse, to update the lexicon, and to edit the input string; strategies which the user can put into effect when he judges repairs too difficult to make. A parser will also occasionally be confronted with *input it cannot handle*. Our approach to knowledge acquisition invites the user to intervene in such cases. Finally, the editor should also be prepared to handle *fragments*. Our experience has been that parsing often fails just short of assembling a final few fragmentary substructures into a single parse tree. The editor should let the user perform this final step. The topic of salvaging incorrect parses is an important one in real-world systems and we plan to devote considerable attention to it.

The Editor as a System that Learns

In the previous subsection the user told the system how to correct a misattached PP. This sort of information could be saved and used by the editor and the parser to correct processing in the future. DIPETT could thereby improve its performance as it accumulates experience in the same way that HAIKU does.

The editor becomes more and more adept at recognizing the most frequent potential errors in each new input as it collects more and more information on tree repairs made by the user. The editor could minimize the editing task by suggesting as potential remedies modifications made in the past to the particular syntactic structures at hand. Suppose that PPs with *since* are often incorrectly attached to a preceding noun instead of a preceding verb. The editor could alert the user to this fact on the basis of its records of PPs with *since*.

Knowledge which the editor acquires about repairs can be generalized into new or changed rules for DIPETT's grammar, subject to the approval of a responsible party. In particular, these modifications could take the form of attachment heuristics telling DIPETT the most likely attachments for certain structurally ambiguous inputs. To continue with the example of a PP with *since*, the editor might generalize a number of instances to propose a heuristic of the form "when

a PP with *since* follows a complement noun phrase, attach it to the preceding main verb". This facility could thus help improve the parser's accuracy.

5.2.3 Refining HAIKU's CA: a Concept Ontology for Noun Semantics

The greatest limitation of HAIKU's current representations of syntactic and semantic patterns —CMPs and CPs—may well be its lack of semantic or conceptual categories for nouns. Such categories would allow HAIKU to make more precise suggestions: it would simply discard candidate patterns associated with nominal fillers with incompatible semantics.

For example, suppose we know that a verb's subject (`psubj`) must be a CONSCIOUS-BEING and its object (`pobj`) a NONDECOMPOSABLE-OBJECT (Bateman *et al.* 1990). HAIKU's CMP entry for the verb would then contain `psubj`/CONSCIOUS-BEING and `pobj`/NONDECOMPOSABLE-OBJECT rather than just `psubj` and `pobj`. Assuming that all noun entries were augmented with the appropriate category or categories, semantic analysis could associate the appropriate category for its filler with each Case marker. Adding this information to CPs would produce items like AGT/CONSCIOUS-BEING and OBJ/NONDECOMPOSABLE-OBJECT instead of just AGT and OBJ. HAIKU could then pass candidate CMPs or CPs through a filter of semantic categories and eliminate the nonsensical.

Consider the semantic categories in Nirenburg & Raskin's (1987) computer science world (CSW). HAIKU currently produces the same `psubj-pobj-adv` CMP and the same AGT-OBJ-TAT CP for "My brother bought a new book yesterday" and "My program compiled a new database yesterday." However, if supplemented with the CSW semantic categories, the first sentence appears as CMP `psubj`/PERSON-`pobj`/DOCUMENT-`adv` and the second as `psubj`/PROGRAM-`pobj`/DATA-`adv`. The respective CPs would be AGT/PERSON-OBJ/DOCUMENT-TAT and AGT/PROGRAM-OBJ/DATA-TAT. Addition of semantic categories has clearly distinguished the two sentences in significant ways: HAIKU would not propose one as an example of the other during Case Analysis.

At this point the reader might be tempted to ask why semantic categories were not added to HAIKU's design at the outset if they have such a positive impact on its processing. There were two reasons for this. First, it is difficult to find an "off-the-shelf" conceptual ontology of the right grain, not so general as to be useless while not so specific as to be tied to the domain. Second, at present ontological knowledge is anything but standard[15]. We believe that using idiosyncratic

---

[15] A recent (November 1992) compilation of "ontological concept-systems", circulated over an electronic news group by Fritz Lehmann, listed 84 different systems!

ontological knowledge would partly contradict our objective of acquiring knowledge by processing text *without* using domain knowledge encoded in advance. Further development and standardization of current ontologies will improve this situation in future. One promising candidate is WordNet (Miller 1990), a lexical network whose hypernymic links we are investigating for use as semantic categories.

Lexical semantics could also help HAIKU disambiguate verb senses. At present all senses that share a particular Case Pattern are stored together. This is in keeping with our view of syntax as the primary carrier of meaning—the store of verb-pattern pairs is *not* intended as a dictionary of verb senses. We simply surmise that, if both Case Marker Patterns and Case Patterns are the same, two senses are close. On the other hand, an ability to capture the subtle differences between verb senses would permit a finer-grained Case Analysis. The effort to accomplish this might outweigh the gain, though, unless the knowledge could be extracted from an existing source like WordNet.

6 Conclusion

To begin processing technical text the TANKA system requires only the syntactic knowledge available in a general purpose English grammar and the lexical knowledge in public domain word resources. Linguistic ambiguities are tentatively resolved by the system and approved or corrected by the user, who also supplies commonsense knowledge. Domain knowledge, the goal of knowledge acquisition in TANKA, is represented by a growing Conceptual Network derived from real texts via analysis on the sentence, clause and noun phrase levels.

This paper focuses on the central Case Analysis module which transforms unconstrained natural language sentences into an intermediate semantic representation used to build this Conceptual Network. Because Case patterns capture many relations among domain objects and processes, they are particularly useful in domains suited to representation by Entity-Relationship models. A mature TANKA system could derive E-R models from text semi-automatically. Another use would be to identify new keywords in text skimming applications. Phrases nearly matching the linguistic expression of existing concepts are likely to mark new concepts which should be added to the domain model. Keywords are generally chosen from such phrases. The deeper its representation of a domain, the more successfully a system could identify the emergence of new concepts in a literature.

Other texts are less well suited to this sort of processing, though. A manual with a high ratio of graphics to text has too many gaps in the presentation for TANKA to process without excessive effort by the user. Furthermore Case patterns in its unconnected text fragments may

refer to objects which only appear in illustrations. Case Analysis will not fail, but it is unlikely to be useful.

Case Analysis will reach a mature state once common Case patterns have been processed. In this state the user will mostly choose amongst plausible alternatives and intervene to correct the system only occasionally. This should occur reasonably soon after use is begun; experience using TANKA suggests that most clauses have simple Case patterns. However, because TANKA will always require some sort of user involvement, we are trying to make interactions as easy and efficient as possible.

Experience to date suggests that Case Analysis is well chosen for our task. HAIKU is proving to be a natural and convenient bridge between the results of parsing and the network that represents a conceptual model of the text.

Acknowledgment

This work has been supported by the Natural Sciences and Engineering Research Council of Canada. Many thanks to the anonymous reviewers for their observations and suggestions, and to Jean-Pierre Corriveau for his comments on the penultimate version of the paper.

References

Aliprandi, G. & Saviozzi, G. (1993), "A Supervised Learning Method to Solve PP-Attachment Ambiguities in Natural Language", *Proc ECML Workshop on Machine Learning and Text Analysis Workshop*, 45-52.

Barker, K., Copeck, T., Delisle, S., & Szpakowicz, S. (1993), "An Empirically Grounded Case System", TR-93-08, Computer Science Department, University of Ottawa.

Barker, K. (1994), "Clause-Level Relationship Analysis in the TANKA System", TR-94-07, Computer Science Department, University of Ottawa.

Basili, R., Pazienza, M. T. & Velardi, P. (1992), "Computational Lexicons: the Neat Examples and the Odd Exemplars", *Proc of 3rd Conf on Applied Natural Language Processing*, 96-103.

Bateman, J. A., Kasper, R. T., Moore, J. D. & Whitney, R. A. (1990), "A General Organization of Knowledge for Natural Language Processing: the Penman Upper Model", TR, Information Sciences Institute, University of Southern California, Marina del Rey, CA.

Blanchon, H. (1992), "A Solution to the Problem of Interactive Disambiguation", *Proc COLING-92*, 1233-1238.

Brent, M. R. (1991a), "Automatic Acquisition of Subcategorization Frames from Untagged Text", *Proc 29th Annual Meeting of the ACL*, 209-214.

Brent, M. R. (1991b), "Automatic Semantic Classification of Verbs from Their Syntactic Contexts: An Implemented Classifier for Stativity", *Proc 5th Conf of the European Chapter of the ACL*, 222-226.

Brown, R. D. & Nirenburg, S. (1990), "Human-Computer Interaction for Semantic Disambiguation", *Proc COLING-90*, 42-47.

Bruce, B. (1975), "Case Systems for Natural Language", *Artificial Intelligence* 6(4), 293-326.

Cardie, C. (1993) "A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis", *Proc AAAI-93*, 798-803.

Celce-Murcia, M. (1975), "Verb Paradigms for Sentence Recognition", *American Journal for Computational Linguistics*, microfiche 38.

Ciravegna, F., Campia, P. & Colognese, A. (1992), "Knowledge Extraction from Texts by SINTESI", *Proc COLING-92*, 1244-1248.

Cohen, W. W. (1990), "Learning from Textbook Knowledge: A Case Study", *Proc AAAI-90*, 743-748.

Cook, W. A. (1979), *Case Grammar: Development of the Matrix Model (1970-1978),* Georgetown University Press.

Copeck, T., Delisle, S., & Szpakowicz, S. (1992), "Parsing and Case Analysis in TANKA", *Proc COLING-92*, 1008-1012.

DeJong, G. (1982), "Automatic Schema Acquisition in a Natural Language Environment", *Proc AAAI-82*, 410-413.

Delisle, S. & Szpakowicz, S. (1991), "A Broad-Coverage Parser for Knowledge Acquisition from Technical Texts", *Proc 5th International Conf on Symbolic and Logical Computing*, 169-183.

Delisle, S. (1990), "Arguments Non-Réalisés Syntaxiquement: Une Analyse pour la Compréhension Automatisée du Langage", *Les Cahiers Linguistiques d'Ottawa*, 19, 1-27.

Delisle, S. (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, January 1994.

Delisle, S., Copeck, T., Szpakowicz, S. & Barker, K. (1993), "Pattern Matching for Case Analysis: A Computational Definition of Closeness". O. Abou-Rabia, C. K. Chang and W. W. Koczkodaj (eds.) *Proc ICCI-93*, 310-315.

Delisle, S., Barker, K., Delannoy, J.-F., Matwin, S. and Szpakowicz, S. (1994), "From Text to Horn Clauses: Combining Linguistic Analysis and Machine Learning". R. Elio (ed.), *Proc 10th Canadian Conf on AI*, 9-16.

DeRose, S. J. (1988), "Grammatical Category Disambiguation by Statistical Optimization", *Computational Linguistics*, 14(1), 31-39.

Di Eugenio, B. (1992), "Understanding Natural Language Instructions: The Case of Purpose Clauses", *Proc 30th Annual Meeting of the ACL*, 120-127.

Dick, J. P. (1992), "A Conceptual, Case-Relation Representation of Text for Intelligent Retrieval", Ph.D. Thesis, TR CSRI-265, Computer Systems Research Institute, University of Toronto.

Dietterich, T. G. (1989), "Machine Learning", *Annual Review of Computer Science*, 4, 1989-1990, 255-306.

Dolan, W., Vanderwende, L., & Richardson, S. (1993), "Automatically Deriving Structured Knowledge Bases from On-line Dictionaries", *Proc 1st Conf of the Pacific Association for Computational Linguistics*, 5-14.

Fass, D. (1989), "Some Connections Between Knowledge Representation and Natural Language Description", TR-89-05, CSS/LCCR , Simon Fraser University.

Fillmore, C. (1968), "The Case for Case" In E. Bach and R. T. Harms (eds.), *Universals in Linguistic Theory*, Holt, Rinehart and Winston.

Gao, Y. & Salveter, S. (1991), "The Automated Knowledge Engineer: Natural Language Knowledge Acquisition for Expert Systems", *Proc 6th AAA Workshop on Knowledge Acquisition for Knowledge-Based System*. 8.1-8.16.

Gomez, F. (1989), "Knowledge Acquisition from Natural Language for Expert Systems Based on Classification Problem-Solving Methods", *Proc 4th AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems*, 15.1-15.18.

Grimes, J. (1975), *The Thread of Discourse,* Mouton.

Grishman, R. & Sterling, J. (1992), "Acquisition of Selectional Patterns", *Proc COLING-92*, 658-664.

Hauptmann, A. G. (1993), "Meaning from Structure in Natural Language Interfaces", Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University.

Jacobs, P. S. (1992), "Joining Statistics with NLP for Text Categorization", *Proc 3rd Conf on Applied Natural Language Processing*, 178-185.

Kaplan, R. M. (1989), "Acquiring Knowledge from Text Using Multiple Methodologies to Accomplish Text Understanding", Ph.D. Thesis, Department of Computer Science, Temple University.

Kim, J.-T. & Moldovan, D. I. (1993), "Acquisition of Semantic Patterns for Information Extraction from Corpora", *Proc 9th IEEE Conf on Artificial Intelligence Applications*, 171-176.

Lang, F.-M. & Hirschman, L. (1988), "Improved Portability and Parsing Through Interactive Acquisition of Semantic Information", *Proc 2nd Conf on Applied Natural Language Processing*, 49-57.

Larson, M. (1984), *Meaning-Based Translation: A Guide to Cross-language Equivalence,* University Press of America.

Liu, R.-L. and Soo, V.-W. (1993), "An Empirical Study on Thematic Knowledge Acquisition Based on Syntactic Clues and Heuristics", *Proc 31st Annual Meeting of the ACL*, 243-250.

Loewen, V. (1988), "ORDINOTRAD: A Machine Translation System based on Case Grammar", M.A. Thesis, School of Translators and Interpreters, University of Ottawa.

Matwin, S. and Szpakowicz, S. (1992), "Machine Learning Techniques in Knowledge Acquisition from Text". *THINK*, vol. 1(2), 37-50.

Matwin, S. and Szpakowicz, S. (1993), "Text Analysis: How Can Machine Learning Help?". *Proc 1st Conf of the Pacific Association for Computational Linguistics*, 33-42.

Miller, G. A. (1990), (eds.), "WordNet: An On-Line Lexical Database", *International Journal of Lexicography*, 3(4).

Montemagni, S. & Vanderwende, L. (1992), "Structural Patterns vs. String Patterns for Extracting Semantic Information from Dictionaries", *Proc COLING-92*, 546-552.

Mooney, R. J. (1990), *A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding*, Morgan Kaufmann Publishers.

Moulin, B. & Rousseau, D. (1992), "Automated Knowledge Acquisition from Regulatory Texts", *IEEE Expert,* October 1992, 27-35.

Nirenburg, S. & Raskin, V. (1987), "The Subworld Concept Lexicon and the Lexicon Management System", *Computational Linguistics,* 13(3-4), 276-289.

Powers, D. M. & C. R. Turk (1989), *Machine Learning of Natural Language*, Springer-Verlag.

Quirk, R., Greenbaum, S., Leech, G. & Svartvik, J. (1985). A Comprehe*nsive Grammar of the English Language*, Longman.

Reimer, U. (1990), "Automatic Knowledge Acquisition from Texts: Learning Terminological Knowledge via Text Understanding and Inductive Generalization", *Proc 5th AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems*, 27.1-27.16.

Riloff, E. (1993), "Automatically Constructing a Dictionary for Information Extraction Tasks", *Proc AAAI-93*, 811-816.

Rinaldo, F. J. (1989), "Deriving Rules for Medical Expert Systems Using Natural Language Parsing and Discourse Analysis", Ph.D. Thesis, Department of Computer Science, Illinois Institute of Technology.

Rousselot, F., Migault, B. & Igot, P. (1992), "Elaboration de Techniques d'Analyse Adaptées à la Construction d'une Base de Connaissances", *Proc COLING-92*, 483-489.

Sager, N. (1981), *Natural Language Information Processing: A Computer Grammar of English and its Applications*, Addison-Wesley.

Smadja, F. & McKeown, K. (1991), "Using Collocations for Language Generation", *Computational Intelligence* 7(4), 229-239.

Smadja, F. A. (1991), "From N-Grams to Collocations: An Evaluation of XTRACT", *Proc 29th Annual Meeting of the ACL*, 279-284.

Somers, H. L. (1987), *Valency and Case in Computational Linguistics*, Edinburgh University Press.

Sparck Jones, K. & Boguraev, B. K. (1987), "A Note on the Study of Cases", *Computational Linguistics* 13(1-2), 65-68.

Szpakowicz, S. & Koperczak, Z. (1990), "Mixed-Strategy Matching in Conceptual Networks", *Proc 5th International Symposium on Methodologies for Intelligent Systems,* 321-328.

Weizenbaum, J. (1967), "Contextual Understanding by Computers", *Communications of the ACM*, 10 (8), 474-480.

Wu, D. (1993), "An Image-Schematic System of Thematic Roles", *Proc 1st Conf of the Pacific Association for Computational Linguistics*, 323-332.

Yang, L. & Szpakowicz, S. (1991a), "Inheritance in Conceptual Networks", *Proc 6th International Symposium on Methodologies for Intelligent Systems*, 191-202.

Yang, L. & Szpakowicz, S. (1991b), "Planning in Conceptual Networks", in F. Dehne, F. Fiala and W. W. Koczkodaj (eds.), *Proc ICCI-91. Advances in Computing and Information. Lecture Notes in Computer Science,* 497, Springer-Verlag, 669-671.

Yang, L. & Szpakowicz, S. (1994), "Path-Finding in Networks", *Proc ICCI-94*.

Zelle, J. M. & R. Mooney (1993), "ILP Techniques for Learning Semantic Grammars", *Proc IJCAI Workshop on Inductive Logic Programming*, 83-92.

Zernik, U. & Jacobs, P. S. (1990), "Tagging for Learning: Collecting Thematic Relations from Corpus", *Proc COLING-90*, 34-39.

Zernik, U. (1992), "Shipping Department vs. Shipping Pacemakers: Using Thematic Analysis to Improve Tagging Accuracy", *Proc AAAI-92*, 335-342.