A Context-Based Free Text Interpreter

XIAOSHAN PAN AND FRANZ J. KURFESS
Department of Computer Science, Cal Poly, San Luis Obispo, CA 93405

Abstract. This paper proposes the design and implementation of a context-based free text interpreter (CFTI), a computer-based natural language understanding (NLU) system that can handle text as generated and used by humans, within a given context. It takes advantage of tracking the contextual meaning of words and phrases during (and after) the development of an ontology for that context, and subsequently uses this information as a knowledge base for interpretation of free text sentences. The system incorporates components of a computer NLU system based on studies of the human understanding processes. Two existing language tools, Link Grammar and WordNet, are examined and incorporated into the system.

The CFTI system is designed and implemented through the use of an expert system shell, CLIPS 6.20, to demonstrate the capability of interpretation and representation of the meaning of free text sentences when a context model is provided. The resultant CFTI system successfully demonstrates the capability to interpret and represent the meaning of free text sentences based on a relatively small-sized context model.

Keywords: Free text, text interpretation, Natural Language Understanding, ontology, context model, meaning representation, Link Grammar, WordNet

1. Introduction

1.1. Research Question and Motivation

Natural language is a fundamental aspect of human behavior, and a crucial component of human life. In written form (e.g., books, magazines, papers) it serves as a knowledge repository passed from one generation to the next. In spoken form it serves as a primary tool with which to communicate (Figure 1-1). According to Chauchard (1964), individual intellectual thought is impossible without the use of natural language. The possibility of human society is due in large part to interpersonal communication and coordination enabled by the use of natural language.

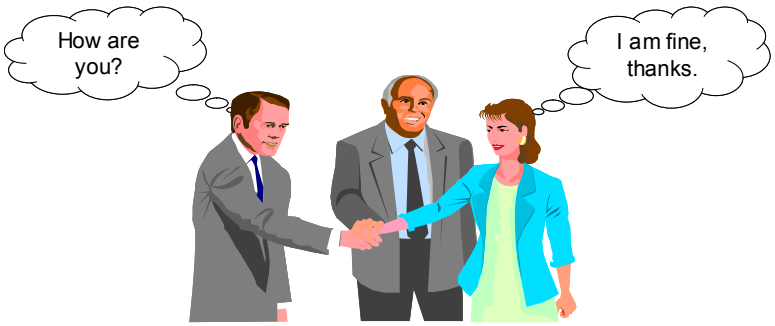


Figure 1-1: Humans communicate via natural language.

Communication among computer systems can utilize formal languages, or protocols, to enable collaboration, information sharing, and problem solving. Utilization of natural language for communication between humans and computer systems, however, remains problematic (Figure 1-2).

The project described here attempts to answer the following question: How can we build a computer system that understands natural language? The field of study on this subject is referred to as Natural Language Understanding (NLU) - a sub-branch of Natural Language Processing (NLP) in Artificial Intelligence (AI). Since the general problem of understanding natural language as used by humans is very difficult for computers, we decided to focus our system on a specific domain through the use of an ontology for that domain. The system is designed to deal with *free text,* i.e. it is not restricted with respect to the language constructs it can handle.

Figure 1-2: It is difficult to communicate with a computer via natural language.

The ability of computers to understand natural language may facilitate their ability to assist human beings in problem solving and decision making. Most of human knowledge is recorded in linguistic form, and therefore computers that can understand natural language could access all this information. Computer systems that search free text documents and present relevant information are examples (i.e., data mining, semantic tagging). Natural language computer interfaces to computers would allow complex systems to be accessible to everyone. Significant advances in NLU have the potential to revolutionize the way computers are used.

1.2. Applications in NLU

Allen (1995) categorizes NLU applications as either text-based or dialogue-based. Text-based NLU applications include:

- search engines which utilize key words as input to find relevant documents in a database (e.g., Citeseer Search at <http://citeseer.nj.nec.com>);
- information retrieval systems which extract information from messages or articles (e.g., a research project by Embley et al. (1998) that extracts information from unstructured documents based on an application ontology that describes a domain of interest); and

- language translators built to translate documents from one language to another (e.g., automobile repair manuals produced in varying languages).

Text-based applications do not normally require human interference. Dialogue-based applications, on the other hand, rely on human-machine communication. Such systems include:

- question-answering systems in which natural language is used to query databases (e.g., START, an MIT natural language question answering system at <http://www.ai.mit.edu/projects/infolab/>, which can answer questions related to geography, arts, and sciences);
- automated customer service systems, many of which provide telephone service (e.g., banking transactions and order catalogue purchasing), and others which provide Internet services (e.g., Dell's auto-email-reply system at <http://support.dell.com/us/en/dellcare.asp> which provides automatic email analysis and response to customer queries); and
- tutoring systems in which machines interact with students (e.g., automated mathematics tutorial systems).

While a significant quantity of applications that utilize NLU have been developed, NLU techniques have yet to reach a mature state of development. Zaenen and Uszkoreit (1996) state that "Despite over three decades of research effort, no practical domain-independent parser of unrestricted text has been developed" (p.110). Ballim et al. (1999) claim that "during these last two years no real improvements have been done in achieving full robustness for a NLP system" (p.2). Some language applications employ annotated phrase-structure grammars (i.e., grammars with hundreds or thousands of rules to describe differing phrase types). Such grammars become difficult to maintain, extend, and reuse as they attain significant size (Zaenen and Uszkoreit, 1996). Statistical parse decision techniques are another approach adopted by some language applications, and require discriminating models of context, which in turn rely on annotated training material to adequately estimate model parameters, rendering its development relatively labor-intensive.

1.3. Project Goal

Ontology construction is an approach to computer domain knowledge representation. An ontology may be defined as specification of a representational vocabulary for a shared domain of discourse which may include definitions of classes, relations, functions and other objects (Gruber, 1993). An ontology includes a selection of specific vocabularies for domain knowledge model construction, and the context of each vocabulary is represented and constrained by the ontology. The meaning of a word in an ontology refers only to the context declared by the ontology.

This project defines *free text* as a text string in the form of an English sentence, consistent with the syntax of English grammar. Its contextual meaning is generally represented by an ontological model (i.e., creation, modification, and deletion of objects and relationships in an object model). In other words, words in such a text string have relationships to the words in the ontology, or context model.

The project proposes the construction of a context-based free text interpreter (CFTI) computer software system that takes advantage of tracking contextual meanings of words

and phrases during (and after) development of an ontology, and subsequently uses this information as the knowledge base for the interpretation of free text sentences. The term "context-based" is included in the title of the system because ontological specification is crucial to the process.

The project includes the following objectives:

- to research and identify essential components of NLU systems, resulting in a theoretical product that leads to the development of CFTI;
- to research existing tools in the field of NLU, and to select and integrate one or more into CFTI if appropriate; and
- to develop the proposed CFTI system in order to demonstrate strategies of extraction and representation of information from free text sentences when a context model is provided.

The approach suggested by the research relies on tracking mapping-relationships between a natural language and a context model, which demands a tool to bolster both pattern matching techniques, and object-oriented design. The C Language Integrated Production System (CLIPS) Version 6.20 (NASA, 2002) supports procedural programming, rule-based programming, and object-oriented design, and has been chosen as a primary tool for the project.

2. Mapping a Natural Language into a Context Model

2.1 Language and Thought

One of the primary uses of language is to express thoughts as they occur in the human mind. These thoughts often are translated into sentences of a language, and expressed through speech or writing.

2.1.1 Real World and Model World

Minsky (1985) states that "we never actually make any direct contact with the outside world. Instead, we work with models of the world that we build inside our brains" (p.110). It is suggested that there exist internal, model worlds within the minds of individuals, in addition to the external, real world (Figure 2-1). Individuals rely on these internal models to reason and react in the real world.

Most people agree that objects exist in the real world, and that these objects are the components of which the world is made. Some may wonder about the nature of the components of which model worlds in human brains are composed.

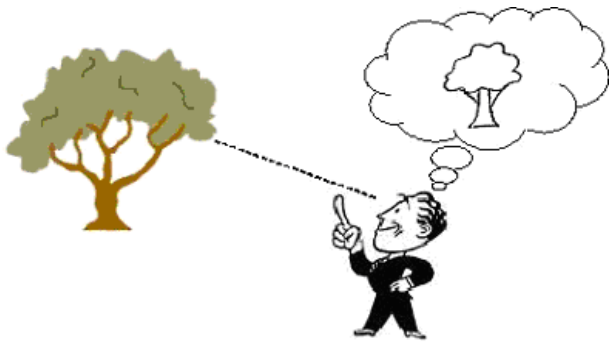


Figure 2-1: Building a model world from the real world.

Because brains cannot hold an object physically (e.g., a real tree), it is a common assumption that brains create representations of real world objects, and store these representations internally in model worlds. All information related to these real world objects is subsequently associated with their representations in the model world. Such representations may be referred to as concepts (e.g., a concept of a tree, or a concept of a chair), and individual thoughts are developed on collections of such concepts. During communication these concepts manifest themselves as words and relationships in the form of natural language (Figure 2-2).

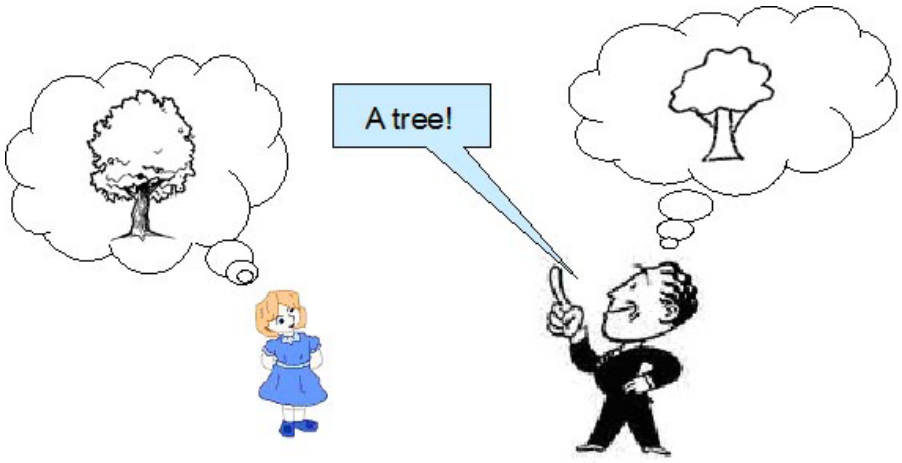


Figure 2-2: Natural language as representation of the world model.

2.1.2 The Role of Natural Language

Natural language plays two principal roles: in the brain, it provides primary building blocks of thought, and in social environments it serves as a tool with which real or model worlds are represented in order to exchange knowledge.

"Thought forms language, and in turn is formed by language" (Chauchard, 1964, p. 7). Initially, the thoughts of young children may not be composed entirely of language elements. Thought manifests itself as language subsequent to the development of language skills. Chauchard (1964) claims that "the adult's thought process is entirely dependent upon the language he has learned as a child" (p. 35).

Chauchard (1964) explains that "whatever thoughts come into a man's mind originate and exist only on the basis of the language's materials, on the basis of its terms and phrases. There is no naked thoughts independent of the language's material (p. 7)".

Speech and writing reflect thought, and thought, is primarily in terms of natural language. Natural language is utilized not only for communication, but also for thinking. For example, "the child talks to himself as though he were thinking aloud" (Piaget, 1959, p. 9).

A group of people who live in relatively close proximity, and speak a common language, may demonstrate relationships among the real world, the model world, and natural language:

1). All individuals live in the same physical world.

2). Each individual builds a model of the physical world in his or her mind. Even though these models vary among individuals, they share commonalities because they all reflect the same physical world (Figure 2-3).

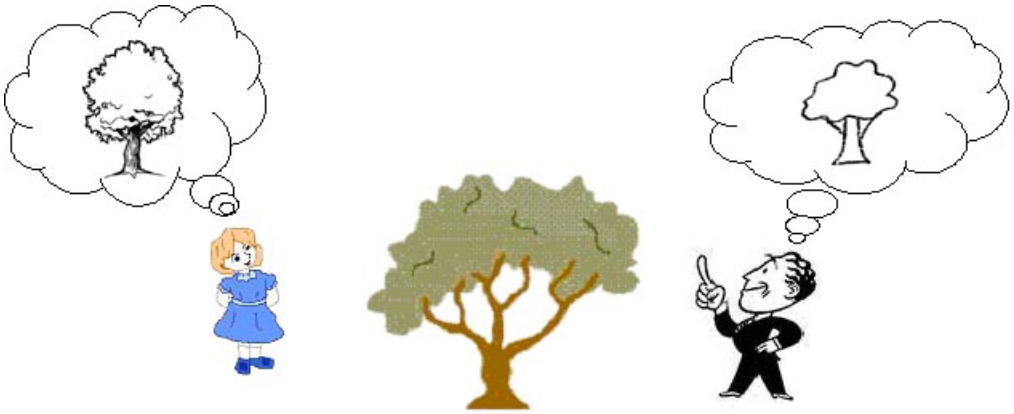


Figure 2-3: Each individual builds a model of the physical world.

3). Model world representations that form human thought manifest themselves as natural language (i.e., words and relationships, see Figure 2-4).

4). Individuals describe their own models, and perceive the models of other individuals via natural language. Understanding among individuals is possible because the various models reflect the same physical world (Figure 2-5).

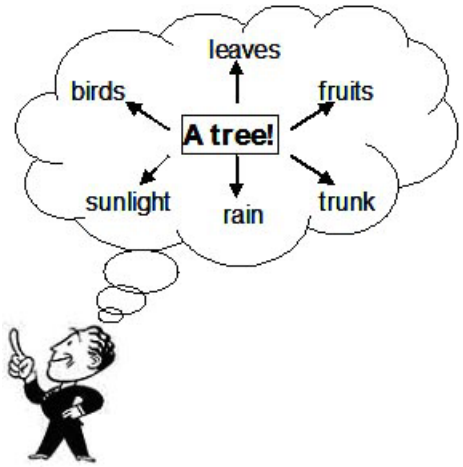


Figure 2-4: Model representations are mostly in the form of languages.

Difficulties in communication may occur among people from different physical environments who share a common natural language, as common words may refer to disparate objects or concepts in their respective physical worlds.

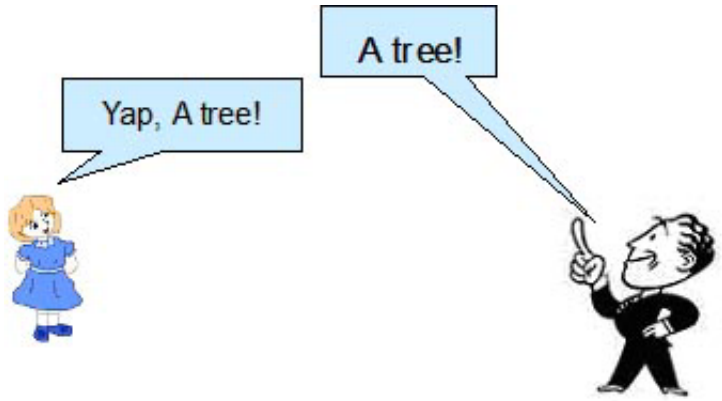


Figure 2-5: Natural language represents the real world for communication among humans.

In the authors' opinion, because thoughts of an adult are primarily in the form of language material, there must exist straight mapping relationships between natural language and the adult's internal model (Figure 2-6).

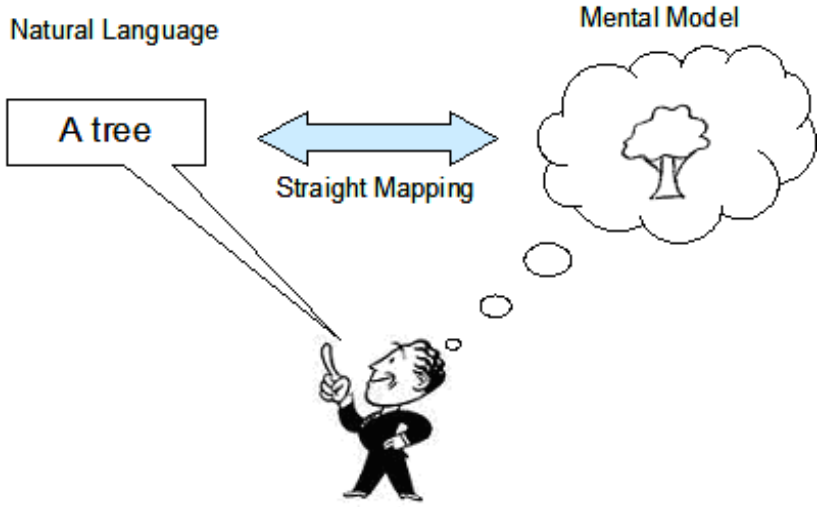


Figure 2-6: There exist straight mapping relationships between natural language and an adult's mental model.

2.2. Ontology and Context Model

As discussed in the beginning of this chapter, the human mind does not interact directly with the real world but rather with models of the real world built and maintained in the brain. Similarly, ontology-based computer systems do not interact directly with the real world but rather with internal models of the real world (Figure 2-7). Such models represent problem domains, and the development of such models in computers is referred to as ontology building.

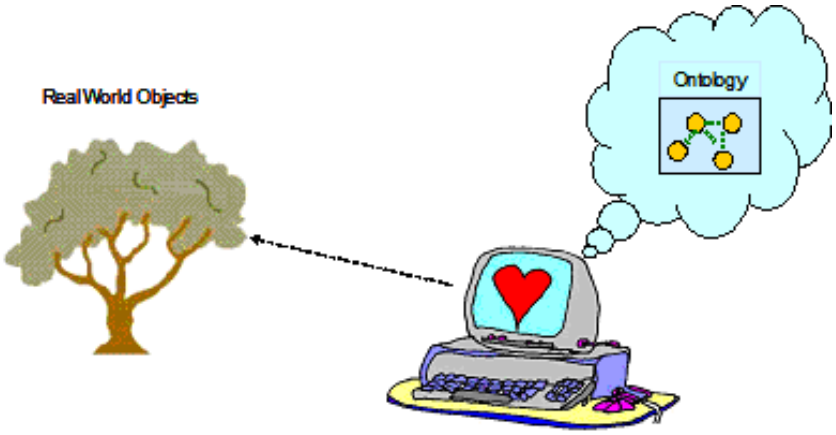


Figure 2-7: An ontology-based computer system works with an internal model of the world.

"In philosophy, ontology is the study of the kinds of things that exist" (Chandrasekaran et al., 1999, p. 20). Ontologies are referred to as content theories in the field of artificial intelligence (AI), and an ontological analysis clarifies the structure of knowledge.

"Ontologies are quintessentially content theories, because their main contribution is to identify specific classes of objects and relations that exist in some domain… Given a domain, its ontology forms the heart of any system of knowledge representation for that domain. Without ontologies, or the conceptualizations that underlie knowledge, there cannot be a vocabulary for representing knowledge (Chandrasekaran et al., 1999, p. 21)".

An ontology serves as a representation vocabulary that provides a set of terms with which to describe the facts in some domain. Concepts represented by an ontology can usually be clearly depicted by a natural language because the ontology and the natural language function similarly (i.e., describing the world). Most vocabularies used in ontologies are direct subsets of natural languages. For example, a general ontology uses "thing", "entity", and "physical"; a specific ontology uses "tank", "weapon", and "tree". Depending on the construction of the ontology, the meaning of those words in the ontology could remain the same as in natural language, or vary completely.

The meaning of ontological terms that are not derived directly from a natural language can still be captured by a natural language. For example, the word "ALLFRD" is used in the IMMACCS ontology (an ontology developed in the Collaborative Agent Design Research Center, Cal Poly San Luis Obispo, California), and means "friendly force" in English.

This project therefore proposes that a computer system able to capture corresponding relationships among vocabularies in its ontology and natural language vocabularies may be able to interpret free text when it is applied to the ontology.

Context is defined by The American Heritage Dictionary (2000) as follows:

"The part of a text or statement that surrounds a particular word or passage and determines its meaning".

In a natural language, a word may have multiple meanings depending on the applicable context. In a computer system, context may be represented and constrained by

an ontology. Vocabularies used in an ontology refer only to the context declared by the ontology. In other words, an ontology provides a context for the vocabulary it contains. Therefore, an ontological model can effectively disambiguate meanings of words from free text sentences. Hence this project considers an ontological model as a context model.

2.3. Computer Understanding

"How do we ever understand anything? Almost always, I think, by using one or another kind of analogy – that is, by representing each new thing as though it resembles something we already know (Minsky, 1985, p. 57)".

Humans are only able to understand what is represented by the model world in their brains; things that could not be represented by the model will not be understood. For example, a child's lack of understanding of advanced mathematics may be a result of insufficient model world elements which represent advanced mathematics, as opposed to intellectual inability.

The same principle applies to a NLU system – the system can only understand things that are represented by its context model. A process of understanding free text is a process of constructing a representation of the meaning(s) carried by the text.

A NLU system which contains a highly advanced context model on the subject of medicine, for example, though able to interpret sophisticated medical conversations, may be unable to interpret simple (from a human perspective) sentences related to sports or cooking due to the lack of concept representation outside the medical domain (Figure 2-8).

A NLU system may be able to understand concepts outside of its context model if the model is extensible (if combined with approaches such as Extensible Ontology and Machine Learning). However, those approaches are not pursued by this research, and a context model in this project is assumed to be static.

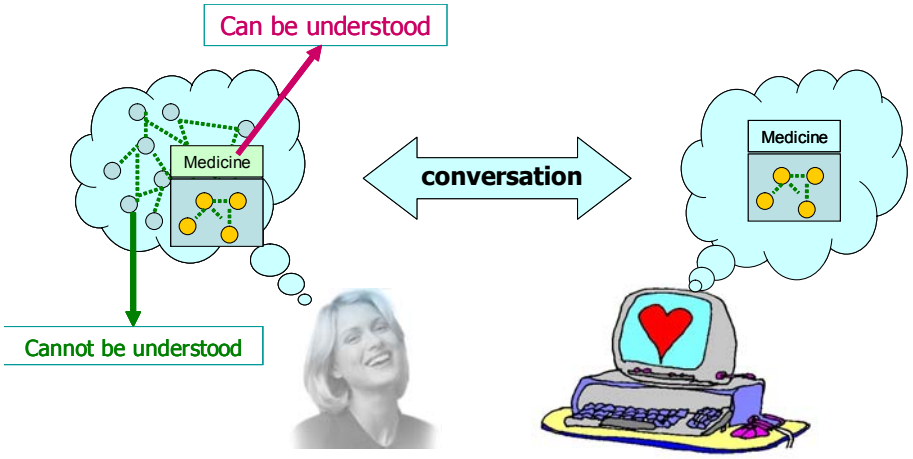


Figure 2-8: A system can only understand things that are representable by its model.

Regardless of whether we are concerned with a dynamic or static model, the quality and structure of the model are crucial in a NLU system. A good context model should be built "with good coherence and stability, and which is able to resist inconsistencies and

ambiguities" (Baud et al. 1993, p. 6). A poorly developed model may lead to unfavorable system performance.

2.4. Chapter Summary

Humans think about the real world mostly in terms of natural language. In AI, ontologies are developed by humans as models of the world for use in computers. A process of understanding free text is a process of model representation of the meaning carried by the text. A NLU system can only understand things that are representable. Direct and indirect mapping relationships exist among vocabularies used by an ontology and vocabularies in a natural language (Figure 2-9); capture and utilization of these relationships is key to development of a NLU system. The quality of the interpretation of free text is strongly dependent on the quality of the model. Coherence, stability, and resistance to inconsistency and ambiguity are desirable ontological model characteristics.

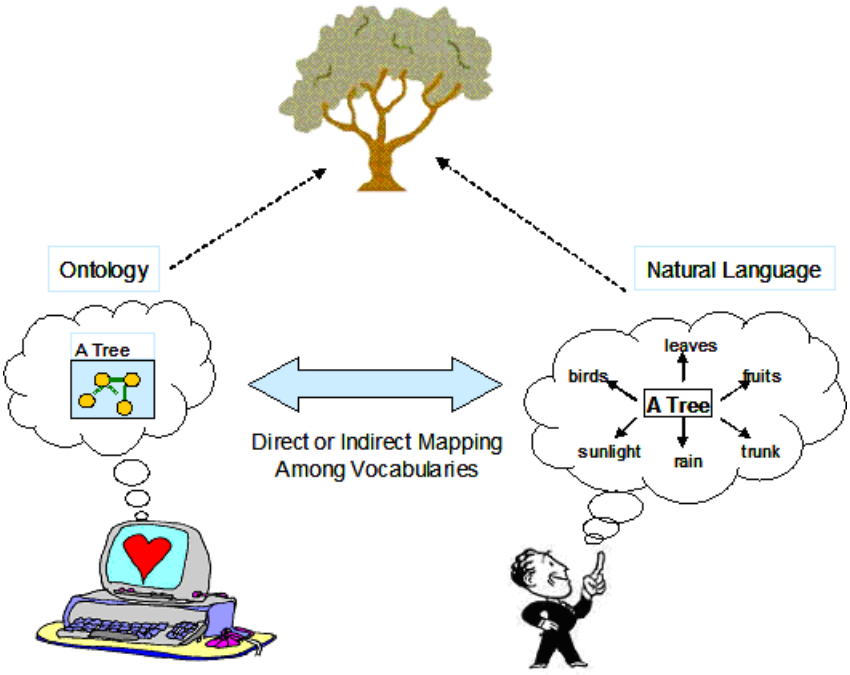


Figure 2-9: Direct and indirect mapping relationships exist among vocabularies in an ontology and a natural language.

3. Context-Based Free Text Interpreter (CFTI) Design

"Part of what a sentence means depends upon its separate words, and part depends on how those words are arranged (Minsky, 1985, p. 266)".

Linguistically, humans combine understanding of relatively small textual units in order to understand larger textual units, guided by syntactic and semantic rules. Syntax relates to arrangement, and semantic to the meaning of words. Similarly, it is necessary

for a NLU system to be able to address syntactic and semantic aspects of natural language.

Chapter Three introduces two existing language tools (i.e., Link Grammar Parser and WordNet database), and proposes the design of a context-based free text interpreter (CFTI).

3.1. Link Grammar Parser

Natural language syntax affects the meaning of words and sentences. The very same words can have different meanings when arranged differently. For example: "a woman, without her man, is nothing" and "a woman: without her, man is nothing" (http://www.p6c.com/joke of the week.html).

The Link Grammar Parser described below was found to be a very effective syntactic parsers, and is therefore incorporated into the design of the CFTI.

3.1.1. Functions of Link Grammar Parser

The Link Grammar Parser, developed at Carnegie Mellon University, is based on "link grammars", an original theory of English syntax (Sleator and Temperley, 1991). The parser assigns to a given sentence a valid syntactic structure, which consists of a set of labeled links connecting pairs of words.

The Link Grammar Parser utilizes a dictionary of approximately 60,000 word forms, which comprises a significant variety of syntactic constructions, including many considered rare and/or idiomatic. The parser is robust; it can disregard unrecognizable portions of sentences, and assign structures to recognized portions. It is able to intelligently guess, from context and spelling, probable syntactic categories of unknown words. It has knowledge of capitalization, numeric expressions, and a variety of punctuation symbols.

3.1.2. Basic Concepts of Link Grammar

The basis of the theory of Link Grammar is planarity, described by Melcuk (1988), as a phenomenon, evident in most sentences of most natural languages, in which arcs drawn to connect words with specified relationships within sentences do not cross. Planarity is defined in Link Grammar as "the links are drawn above the sentence and do not cross" (Sleator and Temperley, 1991, p. 7).

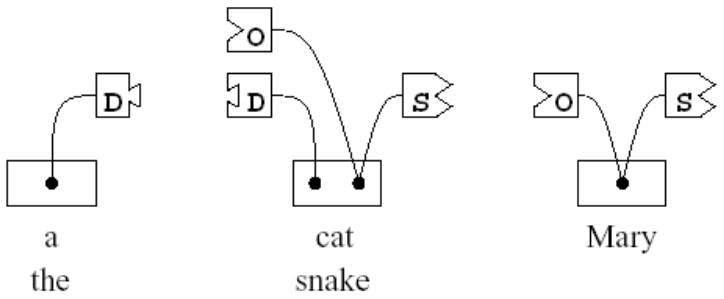


Figure 3-1: Each word is a block with connectors (Sleator and Temperley, 1991, p.3).

"Think of words as blocks with connectors coming out. There are different types of connectors; connectors may also point to the right or to the left… A valid sentence is one in which all the words present are used in a way which is valid according to their rules, and which also satisfies certain global rules (Temperley, et al. 1999, p. 1)".

Each word, from a Link Grammar perspective, is a block with connectors (see Figure 3-1).

Each intricately shaped, labeled box is a connector. A connector is 'satisfied' when it is 'plugged into' a compatible connector (as indicated by shape). A valid sentence is one in which all blocks are connected without a cross. An example of a valid sentence is "the cat chased a snake" (Figure 3-2).

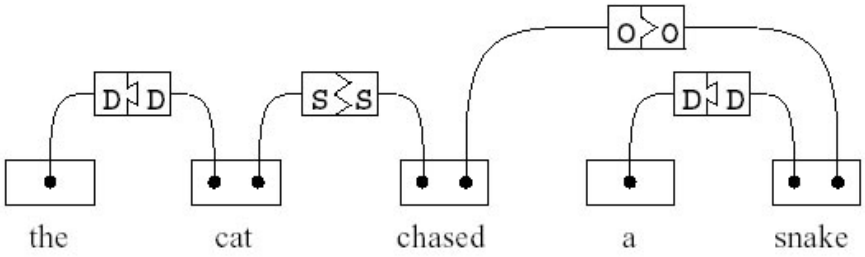


Figure 3-2: A valid sentence contains blocks connected without a cross (Sleator and Temperley, 1991, p.3).

An example of an invalid sentence is "the Mary chased cat", which contains a cross Figure 3-3).

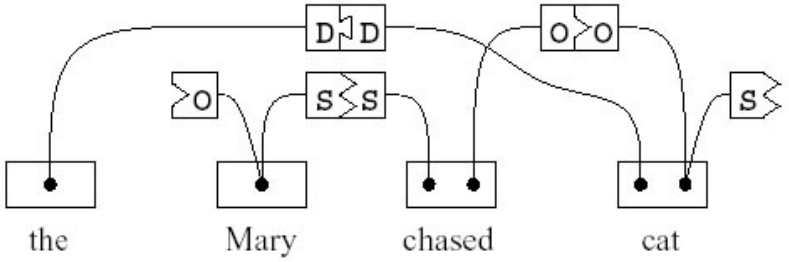


Figure 3-3: An invalid sentence contains blocks connected with crosses (Sleator and Temperley, 1991, p.4).

The Link Grammar Parser finds out all valid linkages within a free text input, and outputs them as grammatical trees. For example, an input such as "The brown fox jumped over that lazy dog" results in the output shown in Figure 3-4:

```
    +------------------------------Xp-------------------------------+
     +---------Wd--------+               +--------Js--------+     |
     |       +-----Ds-----+               |      +-----Dsu----+     |
     |       |     +---A--+---Ss--+--MVp-+      |       +--A--+     |
     |       |     |      |       |      |      |       |     |     |
LEFT-WALL the brown.a fox.n jumped.v over that.d lazy.a dog.n .

Constituent tree:

(S (NP The brown fox)
   (VP jumped
       (PP over
           (NP that lazy dog)))
   .)
```

Figure 3-4: An output produced by the Link Grammar Parser.

3.2. Semantic Knowledge

"Semantic knowledge concerns what words mean and how these meanings combine in sentences to form sentence meanings (Allen, 1995, p. 10)".

Two types of semantic knowledge are essential in a NLU system: 1). lexical knowledge among words despite context (e.g., "children" as the plural form of "child", and the synonym relationship between "helicopter" and "whirlybird"); and 2). contextual knowledge (i.e., how meanings are refined when applied to a specified context).

In CFTI, lexical knowledge is acquired through integration of the system with the WordNet database, and contextual knowledge is acquired by tracking contextual meanings of words and phrases during and after development of an ontology (i.e., context model).

3.2.1. WordNet Database

"WordNet, an electronic lexical database, is considered to be the most important resource available to researchers in computational linguistics, text analysis, and many related areas (Fellbaum, 1999, preface)".

WordNet has been developed since 1985 by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller. Its design is "…inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, and adjectives are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets." (Miller, 1990, p. 1)

The most basic semantic relationship in WordNet is synonymy. Sets of synonyms, referred to as synsets, form the basic building blocks. Each synset has a unique identifier (ID), a specific definition, and relationships (e.g., inheritance, composition, entailment, etc.) with other synsets.

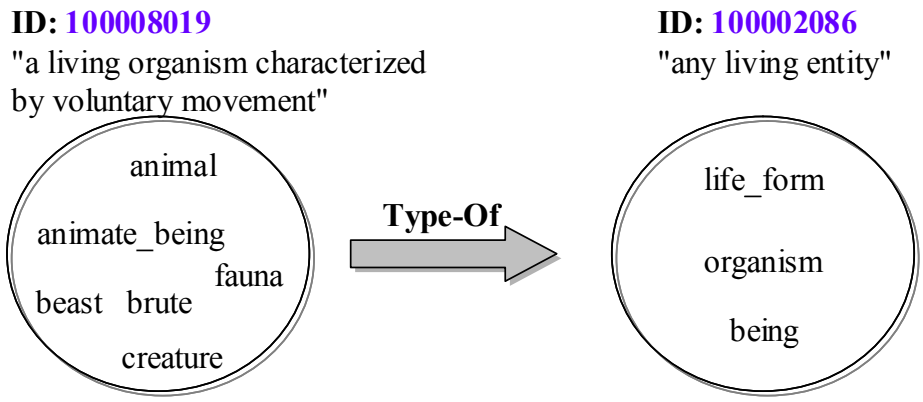


Figure 3-5: Two synsets with a 'type-of' relationship.

Two synsets with a "type-of" relationship are shown in Figure 3-5. The first synset has an ID "100008019", a definition of "a living organism characterized by voluntary movement", and contains six individual words (e.g., "animal", "animate being", etc.). The second synset has an ID "100002086", a definition of "any living entity", and it contains three words (e.g., "life form", "organism", and "being"). The first synset is a "type-of" the second synset.

WordNet contains a significant quantity of information about the English language. It provides meanings of individual words (as does a traditional dictionary), and also provides relationships among words. The latter is particularly useful in linguistic computing. Examples of information provided by the WordNet version 1.7.1 Prolog Database follow:

- A list of all synsets developed to date is provided and includes approximately 174,000 English word entries (Figure 3-6). Each entry contains the synset ID, the word number (i.e., sequence number of the word with the synset), the syntactic category (e.g., noun, verb), the sense number (i.e., a number indicative of the part of speech in which the word is used), and the tag state (i.e., a number which indicates whether or not the sense number is assigned based on frequency of use).

Code Sample

```
...
s(100001740 1 "entity"       n 1  1)
s(100001740 2 "something"    n 1  0)
s(100002086 1 "life form"    n 1  0)
s(100002086 2 "organism"     n 1  1)
s(100002086 3 "being"        n 2  1)
s(100002086 4 "living thing" n 1  1)
s(100002880 1 "life"         n 10 1)
...
```

Figure 3-6: Word entries in WordNet 1.7.1. Prolog database.

- A definition of each synset is provided (Figure 3-7). Each synset is assigned one unique definition; words with multiple meanings appear in multiple synsets.

Code Sample

```
...
g(100001740 "anything having existence (living or nonliving)")
g(100002086 "any living entity")
g(100002880 "living things collectively”)
g(100003011 "a discrete unit of living matter")
...
```

Figure 3-7: Each synset is specified with a gloss.

- “Type-Of” relationships among synsets are described (i.e., the first synset is a “type-of” the second synset indicates an inheritance relationship). (Figure 3-8).

Code Sample

```
...
hyp(100002086 100001740)
hyp(100002880 100002086)
hyp(100003011 100002086)
hyp(100003095 100001740)
...
```

Actual Meaning

```
...
"life form" is a type of "entity"
"life"      is a type of "being"
"biont"     is a type of "living thing"
"cell"      is a type of "entity"
...
```

Figure 3-8: ‘Type-Of’ relationships among synsets.

- “Part-Of” relationships among synsets are provided and, for example, could specify that the first synset is a substance meronym of the second synset (Figure 3-9). This type of relationship applies only to nouns, and is equivalent to the composition relationship as used in an ontology.

Code Sample

```
...
ms(101412507 102623403)
ms(101665957 105640933)
ms(102219585 103485635)
ms(102280809 102280226)
...
```

Actual Meaning

```
...
"eiderdown" is part of "continental quilt"
"oxtail"    is part of "oxtail soup"
"atenolol"  is part of "Tenoretic"
"belting"   is part of "belt"
...
```

Figure 3-9: ‘Part-Of’ relationships among synsets.

- "Derived-From" relationships between individual words are described and, for example, might indicate that the first word is derived from the second word (Figure 3-10).

Code Sample

```
...
per(302468959 2 110432891 1)
per(302468959 1 110432891 1)
per(302469083 2 109418568 1)
per(302469083 1 301386747 1)
...
```

Actual Meaning

```
...
"abatic"  is derived from "abasia"
"abasic"  is derived from "abasia"
"abaxile" is derived from "axis"
"abaxial" is derived from "axial"
...
```

Figure 3-10: 'Derived-From' relationships between individual words.

- "Similar" relationships among synsets are described, applicable to verbs and adjectives (Figure 3-11).

Code Sample

```
...
sim(300003057 300003314)
sim(300003057 300003469)
sim(300003057 300003588)
sim(300003314 300003057)
...
```

Actual Meaning

```
...
"aborning" is similar to "birthing"
"aborning" is similar to "nascent"
"aborning" is similar to "parturient"
"lying-in" is similar to "aborning"
...
```

Figure 3-11: 'Similar' relationships among synsets.

- Semantic "Antonymy" relationships among individual words are described which may in given contexts express opposite meanings (Figure 3-12).

Code Sample

```
...
ant(100124895 1 100125039 1)
ant(100130078 1 100130362 1)
ant(100186549 1 100186422 1)
ant(100195217 1 100195346 1)
...
```

Actual Meaning

```
...
"earned run"       can be an antonym of "unearned run"
"promotion"        can be an antonym of "demotion"
"sitting trot"     can be an antonym of "rising trot"
"domestic flight" can be an antonym of "international flight"
...
```

Figure 3-12: 'Antonymy' relationships among individual words.

Such examples indicate that the content of WordNet exceeds that of traditional dictionaries, and may be used as a the basis for an ontology of the English language.

While WordNet links words and concepts through a variety of semantic relationships based on similarity and contrast, it "does not give any information about the context in which the word forms and senses occur" (Fellbaum, 1999, p. 12). In the CFTI, refinement of word meanings in specific contexts (i.e., contextual knowledge) is accomplished by mapping relationships between natural language and a context model.

3.2.2. Relationships Between Natural Language and Context Model

Ontologies provide context for vocabularies which they contain. Direct and indirect mapping relationships exist among ontological vocabularies and natural language vocabularies. Understanding of such relationships may enable a system to understand contextual meanings of words used in the context defined by an ontology. The application of the same word to other ontologies could produce other meanings.

In practice, a process of tracking mapped relationships between a natural language and a context model is a process of interpretation of the model (i.e., what a model really means) through the use of a natural language. The word "ALLFRD", for example, in the IMMACCS ontology means "friendly force" in English. Contextual knowledge can be attained directly from the ontology designer(s), or can be attained through utilization of an automated process if the ontology design follows formalized conventions. Such an

implementation that captures contextual knowledge through the use of CLIPS 6.20 is discussed In Chapter Four.

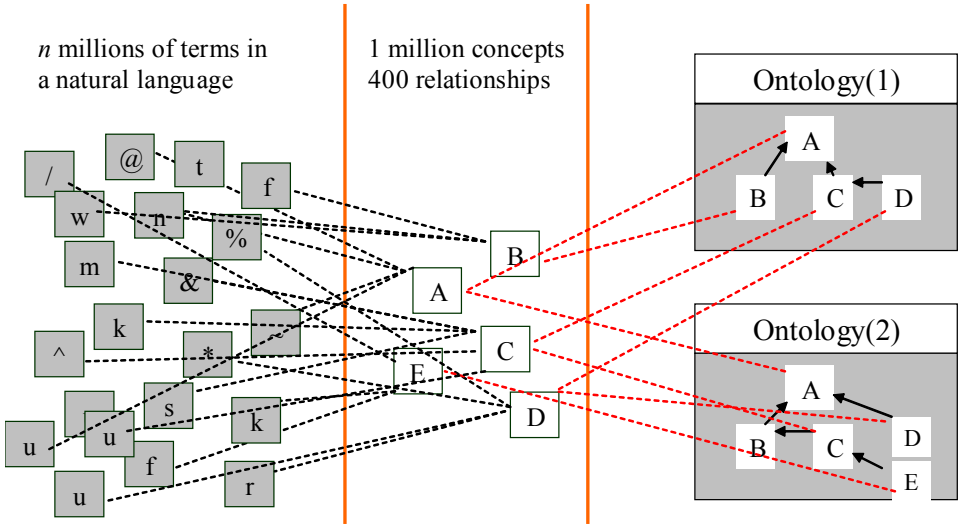


Figure 3-13: Mapping from natural language to context models.

From the perspective of a NLU system which employs appropriate lexical and contextual knowledge, interpretation of a free text sentence is a process of mapping the sentence from natural language to a context model (Figure 3-13). Different context models may produce different results simply because words could have different meanings in different contexts.

3.3 Representation of Meaning

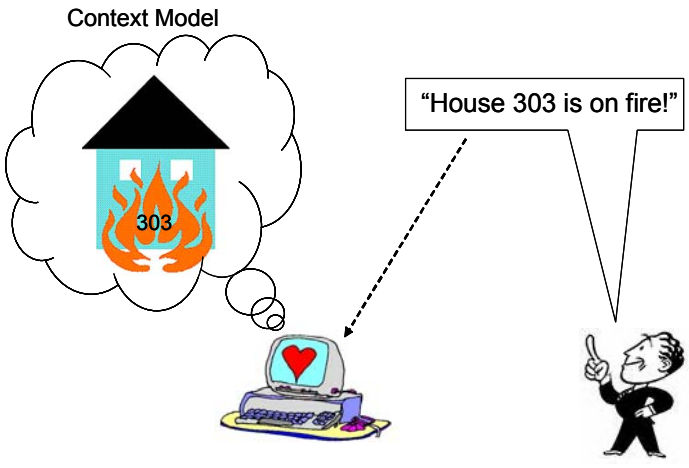


Figure 3-14: Representing the meaning of a sentence in a computer.

Understanding a free text sentence is a process of representing the sentence's meaning through the use of a model internal to an interpreter. This concept is applicable to both humans and computers. In the CFTI, the representation of meaning is accomplished by

manipulations of a context model (i.e., creation, modification, and deletion of objects and relationships in an object model).

For example, a hazard detection system receives a free text sentence "House 303 is on fire!". If the system is able to model this information correctly (i.e., locate the instance of the house in the model and set its attribute to "on fire"), then it is assumed that the system understands the meaning of the sentence (Figure 3-14).

3.4. System Design

In order to interpret a free text sentence correctly, a NLU system needs to conduct the following tasks:

1). Analyze the syntactic structure of the sentence.
2). Analyze the lexical meaning of the words in the sentence.
3). Refine the meanings of the words through the application of a context model.
4). Represent the meaning of the sentence in the model.

Subsequent computations may take place after the above tasks. If an agent engine is attached to the context model, for example, then some agents may react to the changes in the model, if appropriate.

Figure 3-15 illustrates the processing of a free text message by the CFTI system and the subsequent representation in the model.

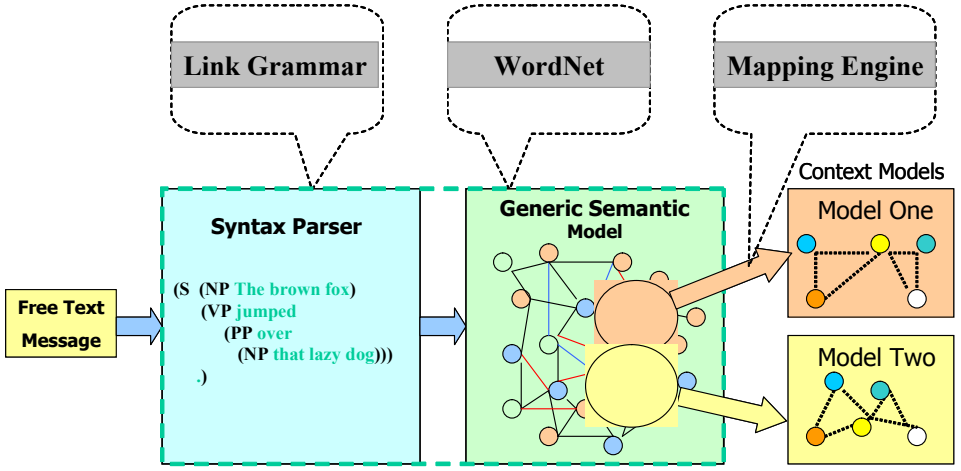


Figure 3-15: From free text messages to context models.

Even though the CFTI system requires an ontological model for the acquisition of contextual knowledge and the representation of meanings, the system is not constrained to any particular knowledge domain. A system change from one ontological model to another does not require significant system reconfigurations beyond the replacement of one ontology with another.

4. Implementation

Chapter Four documents the implementation of the CFTI with CLIPS 6.20, explains the development of each software component, and demonstrates two testing scenarios at the end of the Chapter.

4.1. System Architecture

The architecture of the CFTI system consists of five components: Link Grammar, Lisp Simulator, WordNet database, a mapping engine, and a context model (Figure 4-1).

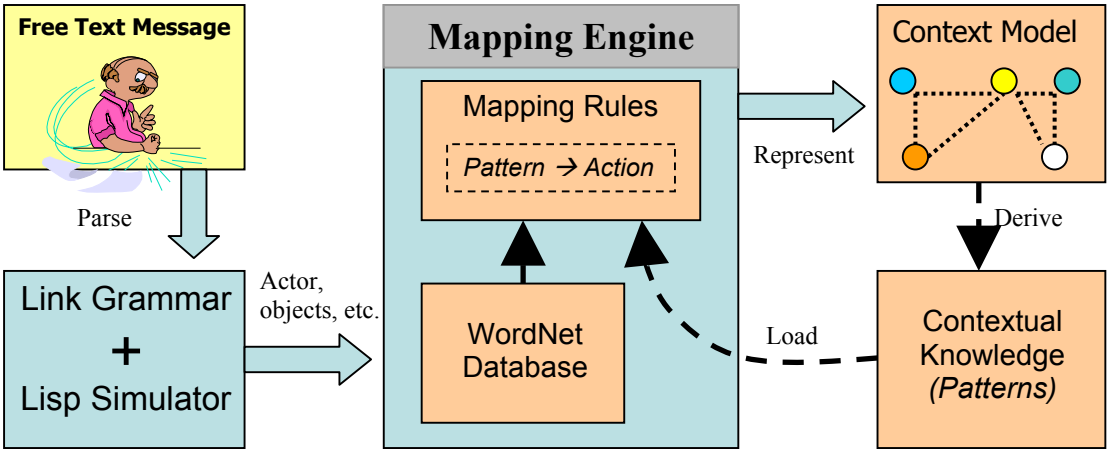


Figure 4-1: CFTI system architecture.

The Link Grammar is integrated into the CFTI as a syntax parser. With a free text sentence as input, it returns a string, which represents the syntactic structure of the sentence.

Lisp Simulator is a software component implemented in CLIPS. It simulates internal functions (e.g., car, cadr, tail recursion, etc.) that are available in the Lisp language but not available in CLIPS. These functions conveniently facilitate CFTI access to syntactic trees (i.e., outcomes from Link Grammar) and were developed and implemented in CLIPS by the authors.

The WordNet database serves CFTI as a lexical knowledge base, which provides relationships between concepts (i.e., synsets) and words.

A mapping engine is implemented in the form of CLIPS rules. It analyzes the outcome of the Link Grammar and establishes meaning based on the knowledge contained in WordNet (lexical knowledge) and a context model (contextual knowledge).

The context model is a formal ontology created in standard UML (i.e., Unified Modeling Language) format and is subsequently converted into CLIPS COOL classes. A body of domain knowledge is thus represented as a group of classes and relationships. The context model is employed by CFTI for contextual knowledge derivation and meaning representation.

4.2. Integration of Link Grammar with CLIPS

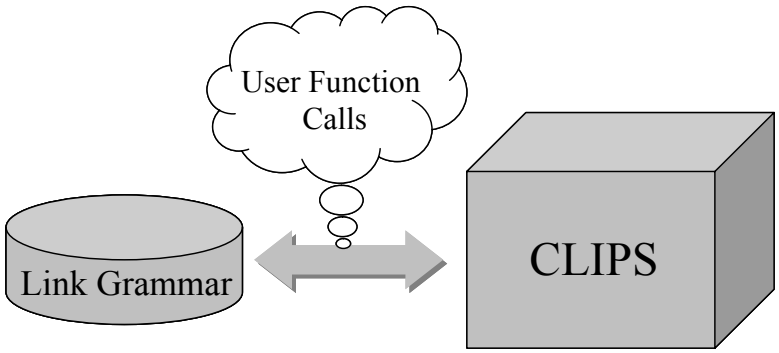


Figure 4-2: Integrating Link Grammar with CLIPS.

One of the most important features of CLIPS is an ability to integrate CLIPS with external functions or applications. CLIPS is written in the C language; user-defined functions are described in CLIPS through modification of a function named *UserFunctions*. Each external function to be integrated with CLIPS is declared in *UserFunctions*, and can thereafter be called from CLIPS (Figure 4-2). A sample code for *UserFunctions* is shown in Figure 4-3, where *clips_parse()* is an external function defined in the Link Grammar Parser.

```
/***********************************/
/* UserFunctions: Notifies CLIPS of */
/*   any user-defined functions.    */
/***********************************/
void UserFunctions()
{
  /*=======================================*/
  /* Declare your C functions if necessary. */
  /*=======================================*/
  extern void clips_parse();

  /*=========================================================*/
  /* Call DefineFunction to register user-defined functions. */
  /*=========================================================*/
  DefineFunction2("parse", 's', PTIF clips_parse, "clips_parse", "11s" );

}
```

Figure 4-3: Sample code of user-defined functions in CLIPS.

The Link Grammar was developed in the C language; the Link Parser API was provided to give users flexibility to integrate the parser with their applications (Temperley et al., 1999). The authors developed a function named *clips_parse* as the interface between the Link Grammar Parser and CLIPS, and subsequently re-compiled all source code as *CLIPWin* executable.

Figure 4-4 demonstrates a successful call to the Link Grammar Parser from the CLIPS shell after integration.

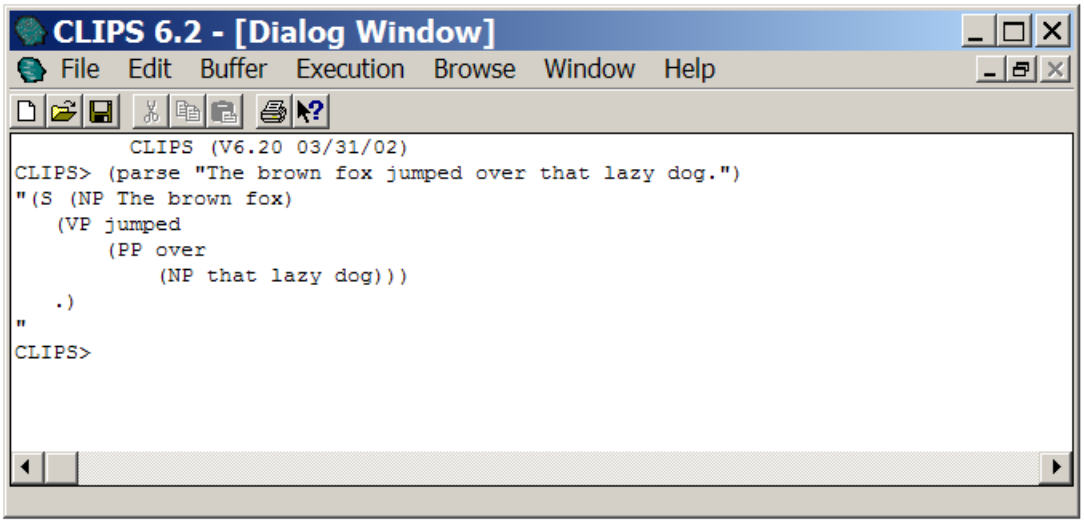


Figure 4-4: An external function call to the link grammar parser from CLIPS.

The following steps take place subsequent to such a call (Figure 4-4):

1). The function ***parse*** receives the original free text string "The brown fox jumped over that lazy dog", then passes this call to the external function ***clips_parse***;

2). the Link Grammar Parser is called through the Link Grammar API by ***clips_parse*** to process the input string. The return value is a string which represents the syntactic structure of the original input; and,

3). the return value is passed back to the original caller ***parse*** in CLIPS.

Although the returned syntactic tree in Figure 4-4 may be meaningful to humans (i.e., reflective of the structure of a sentence), it is meaningless to a computer because it is merely a string (i.e., a primitive data type).

The Lisp Simulator is implemented in CLIPS for syntactic tree access. The CFTI is able to retrieve useful information from a syntactic tree through the use of the Lisp Simulator.

4.1 Lisp Simulator for Tree Access

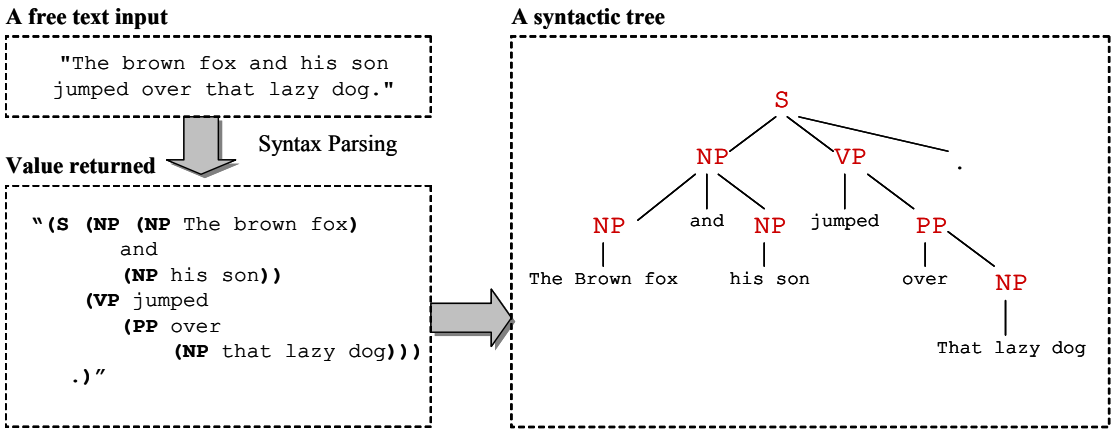


Figure 4-5: The parser returns a string that represents a syntactic tree.

If a free text string is parsed by calling the *parse* function subsequent to integration of the Link Grammar Parser with CLIPS, the system returns another string that represents the syntactic structure of the text (Figure 4-5).

As shown in Figure 4-5, the returned value is a string which contains nested parentheses, potentially problematic for direct CLIPS access, because CLIPS was not built to access nested list structures (e.g., "(a (b c) d)"). Another programming language - Lisp, is more readily capable of such list processing.

The authors developed a software component named Lisp Simulator, which through the use of CLIPS functions simulates the Lisp language and provides basic functions for list processing (e.g., car, cdr, and tail recursion). Figure 4-6 demonstrates some basic Lisp functions simulated in CLIPS.

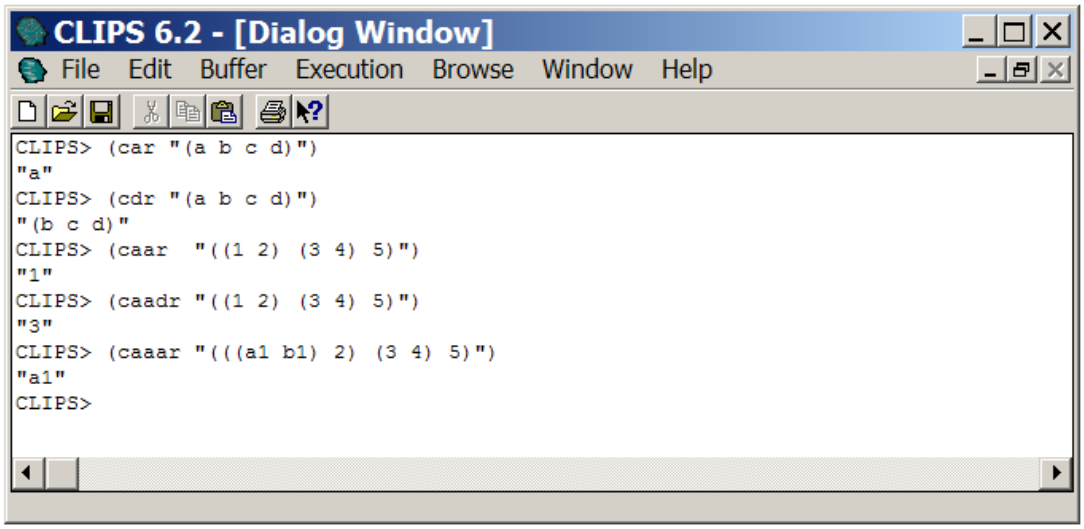


Figure 4-6: Basic Lisp functions simulated in CLIPS.

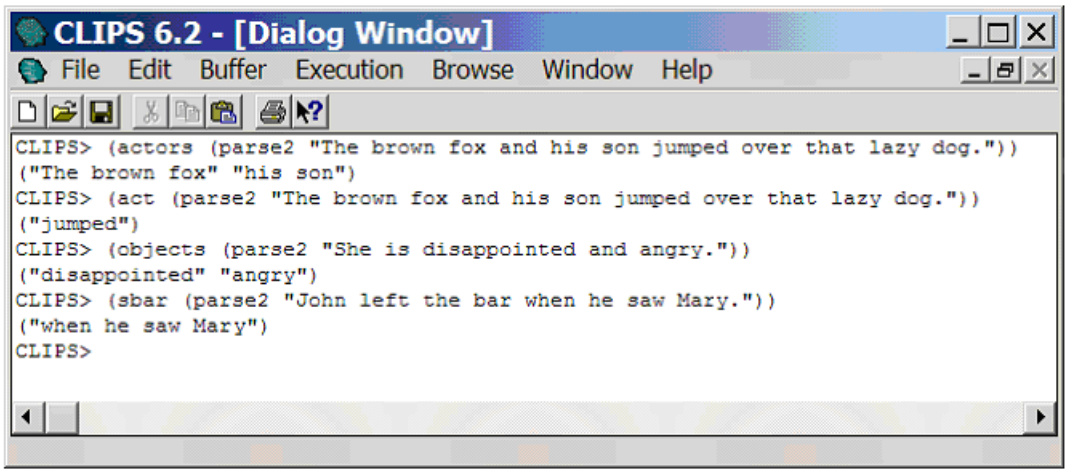


Figure 4-7: Tree access functions.

A set of tree access functions based on the Lisp Simulator are built to retrieve information from a syntactic tree (Figure 4-6). These functions include:

- ***actors***, which looks for the subject(s) of a sentence (e.g., the subjects of "The brown fox and his son jumped over that lazy dog" are "The brown fox" and "his son"),
- ***act***, which looks for the verb(s) of a sentence (e.g., the verb of "The brown fox and his son jumped over that lazy dog" is "jumped").
- ***objects***, which looks for the objects of a sentence (e.g., the objects of "She is disappointed and angry" are "disappointed" and "angry"), and
- ***sbar***, which looks for the decorative clauses of a sentence (e.g., the decorative clause of "John left the bar when he saw Mary" is "when he saw Mary").

Figure 4-7 shows the above examples in the CLIPS 6.20 shell.

4.4. Integration of WordNet Database with CLIPS

The WordNet database provides copious lexical knowledge about the English language. Nouns, verbs, and adjectives are organized into synonym sets, each of which represents one underlying lexical concept. Different relationships link the synonym sets. Two primary types of relationships are chosen for integration into the current implementation of the CFTI: synonymy and irregular forms.

Synonymy is the most basic semantic relation in WordNet; sets of synonyms (synsets) form basic building blocks of the database. The original database is in Prolog format and had to be converted to CLIPS format. Figure 4-8 shows the conversion of the synonymy relationship from Prolog format to CLIPS format.

Prolog Format

```
...
s(100001740 1 "entity"       n 1  1)
s(100001740 2 "something"    n 1  0)
s(100002086 1 "life form"    n 1  0)
s(100002086 2 "organism"     n 1  1)
s(100002086 3 "being"        n 2  1)
s(100002086 4 "living thing" n 1  1)
s(100002880 1 "life"         n 10 1)
...
```

CLIPS Format

```
...
(word 100001740 1 "entity"       n 1  1)
(word 100001740 2 "something"    n 1  0)
(word 100002086 1 "life form"    n 1  0)
(word 100002086 2 "organism"     n 1  1)
(word 100002086 3 "being"        n 2  1)
(word 100002086 4 "living thing" n 1  1)
(word 100002880 1 "life"         n 10 1)
...
```

Figure 4-8: Conversion from Prolog format to CLIPS format.

CLIPS rules thereafter can be built to search for synonymy relationships in free text sentences. Figure 4-9 demonstrates how synonymy relationships are searched by the use of a CLIPS rule.

```
(defrule find-synonyms
   (mg (order ?order)
       (body  ?word1))
   (object (is-a CLASS_NAME)
       (className    ?className)
       (alternatives $? ?key $?))

   (word ?id ?num1 ?word2 $?)
   (word ?id ?num2 ?key $?)
   (test (neq ?num1 ?num2))
   (test (eq  ?word2 (str-cat ?word1)))
=>
   ;ACTIONS
)
```

Figure 4-9: Searching for synonyms by the use of a CLIPS rule.

Irregular word form is another important relation provided by WordNet, which covers irregular forms among nouns (i.e., singular and plural forms), adjectives and verbs (Figure 4-10). Over 12,700 irregular form entries in the WordNet database (version 1.7.1) provide thorough coverage of the language.

```
...
(brief "children"     "child")
(brief "chillies"     "chilli")
(brief "chinaberries" "chinaberry")
...
(brief "prettier"     "pretty")
(brief "prettiest"    "pretty")
(brief "pricier"      "pricy")
(brief "priciest"     "pricy")
...
```

Figure 4-10: Code sample for irregular form relationship.

Searching for irregular form relationships can be accomplished by a CLIPS rule as shown in Figure 4-11.

As discussed in Chapter 3, WordNet is a lexical relational database of the English language, and it does not give any information about the context in which the word forms and senses occur (i.e., contextual knowledge). Contextual knowledge in the CFTI is achieved by utilizing the mapping relations between a natural language and a context model.

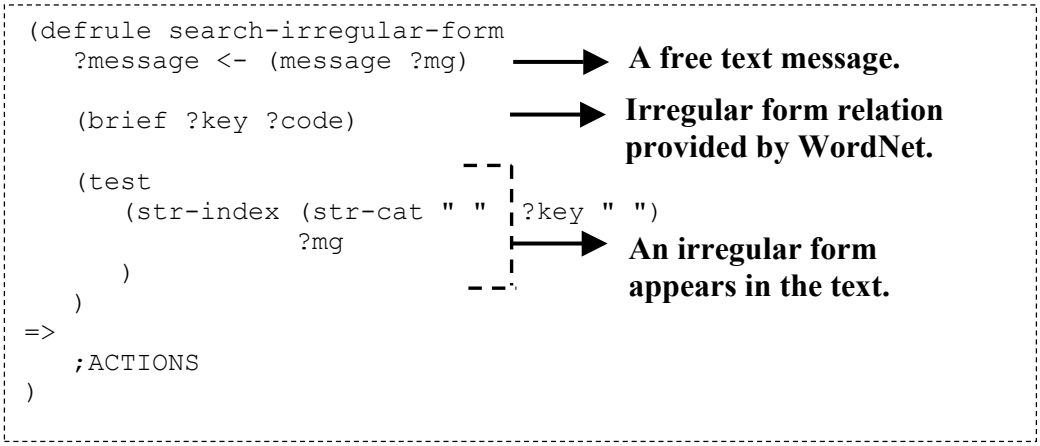


Figure 4-11: Searching for irregular form relationships by the use of a CLIPS rule.

4.5. From a Free Text Sentence to an Object

4.5.1. Representations of a Sentence

> sen·tence n. 1. A grammatical unit that is syntactically independent and has a subject that is expressed or, as in imperative sentences, understood and a predicate that contains at least one finite verb (The American Heritage, 2000).

A simple sentence is a group of words that contains a subject and a predicate, which describes an event, or state(s) of an object (Figure 4-12).

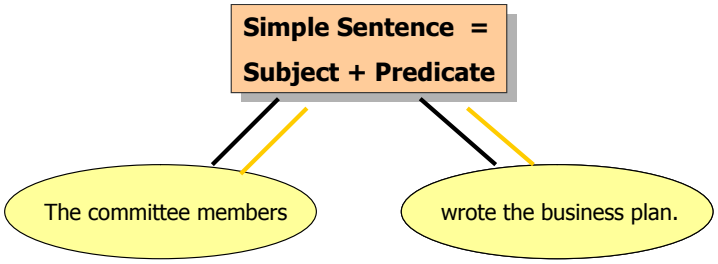


Figure 4-12: A simple sentence contains a subject and a predicate.

In the context of the CFTI, we define a sentence as a group of words containing objects and descriptions of their attributes. Such objects with attributes represent the meaning of a sentence. Hence the tasks of the system are:

- Identification of what the objects are,
- Identification of what the attributes are (relationships are considered to be special type of attribute), and
- Representation of the objects and attributes in a context model.

An object in a sentence may be an object that already exists in a context model, or may be a new object.

Figure 4-13 is a Unified Modeling Language (UML) class diagram employed by the CFTI as an intermediate representation of a sentence. The meaning of a sentence is represented as "CHUNK" objects with associations prior to being mapped into a context model.

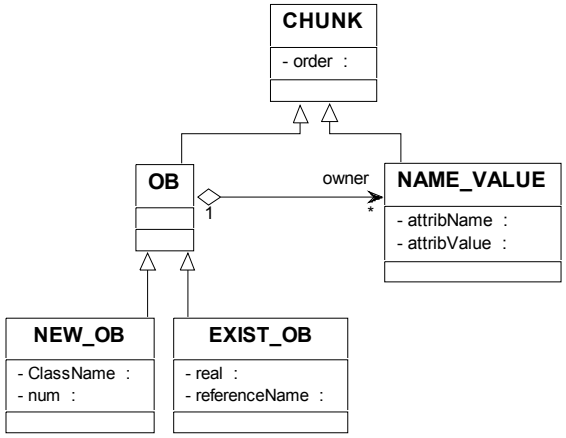


Figure 4-13: An intermediate representation of sentences.

Referring to Figure 4-13, "OB" represents an object contained by a sentence, and can be associated with zero or more "NAME_VALUE" attributes. Each "NAME_VALUE" holds an attribute name and a value associated with that name. There are two types of "OB": "NEW_OB" and "EXIST_OB". "NEW_OB" represents a new object, and "EXIST_OB" represents an object that exists in a context model.

4.5.2. Mapping Techniques

A simple model, Mini-IMMACCS (Figure 4-14), was developed to test and examine techniques which utilize CLIPS 6.20 for mapping from a natural language to a context mode. Mini-IMMACCS contains ontological model components (various classes and differing types of associations) sufficient to reasonably apply conclusions (based on the tests and examination) to other models of larger scale.

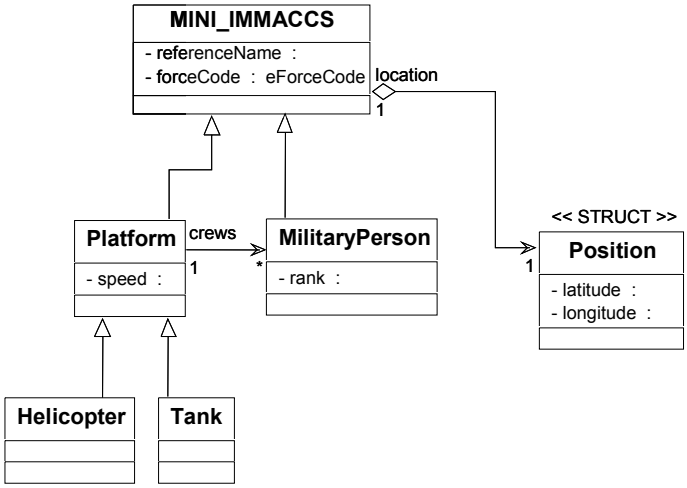


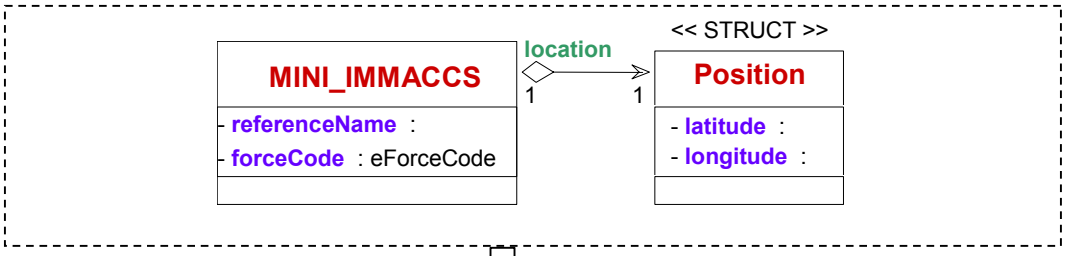
Figure 4-14: Mini-IMMACCS.

4.5.2.1. Types of Symbols and the Design of PATTERN Classes

An ontology serves as a representation vocabulary that provides a set of terms with which to describe facts in some domain. Based on the analysis of Mini-IMMACCS by the use of CLIPS COOL, we categorize symbols (or terms) in an ontology as:

- symbols for class names (e.g., Platform, MilitaryPerson, and Tank),
- symbols for attribute names (e.g., speed, rank, and forceCode), and
- symbols for attribute values (e.g., eForceCode).

Associations between classes are treated as special types of attributes. For example, two classes in the model, MINI_IMMACCS and Position, are represented in CLIPS COOL as *defclass* constructs (Figure 4-15).




```
(defclass MAIN::MINI_IMMACCS
   (is-a USER)
   (role concrete)
   (pattern-match reactive)
   (slot referenceName
      (type STRING)
      (default ?DERIVE)
      (create-accessor read-write))
   (slot forceCode
      (type SYMBOL)
      (allowed-values ALLHOS ALLFRD)
      (default ALLFRD)
      (create-accessor read-write))
   (slot location
      (type INSTANCE-NAME)
      (default [nil])
      (create-accessor read-write)))
```

```
(defclass MAIN::Position
   (is-a STRUCT)
   (role concrete)
   (pattern-match reactive)
   (slot latitude
      (type FLOAT)
      (default ?DERIVE)
      (create-accessor read-write))
   (slot longitude
      (type FLOAT)
      (default ?DERIVE)
      (create-accessor read-write)))
```

Figure 4-15: Examples of CLIPS defclass constructs.

In the CLIPS defclass constructs, symbols used in the ontology fall within the abovementioned categories: 'MINI_IMMACCS' and 'Position' are symbols for class names; 'referenceName', 'forceCode', 'location', 'latitude', and 'longitude' are symbols for attribute names; and 'ALLHOS' and 'ALLFRD' are symbols for attribute values. The symbol 'location' refers to the relationship between the two classes, and is represented as an attribute name in the class MINI_IMMACCS.

PATTERN classes are designed to represent relationships between ontological and natural language vocabularies (Figure 4-16).

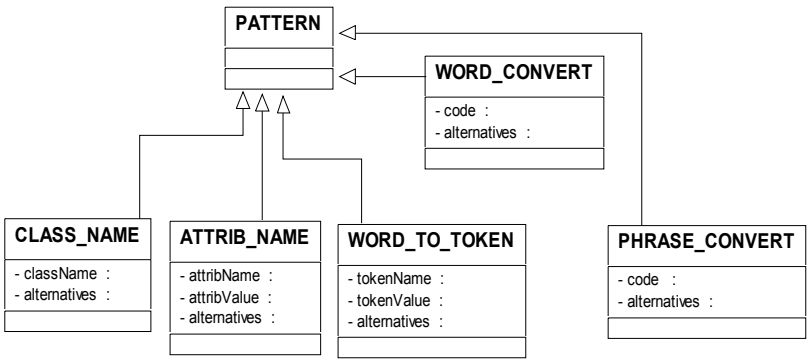


Figure 4-16: The PATTERN classes.

Descriptions of the classes shown in (Figure 4-16) are as follows:

- CLASS_NAME represents relationships between vocabularies used as class names in a context model and vocabularies in a natural language. For example, the symbol 'Tank' refers to 'tank' or 'tanks' in English, a relationship which can be represented by an instance of CLASS_NAME:

  ([className-1] of CLASS_NAME
  (className Tank)
  (alternatives "tank" "tanks"))

  When the word 'tank' or 'tanks' appears in a free text sentence, CFTI can refer to it as Tank object in the Mini-IMMACCS model through the use of a CLIPS rule shown in Figure 4-17.

```
(defrule find-class-name
   (mg  (order ?order)
        (body ?key)
   )
   (object (is-a CLASS_NAME)
        (className  ?className)
        (alternatives $? ?key $?)
   )
=>
   ;actions
)
```

Figure 4-17: A CLIPS rule to search class names from a text string.

- ATTRIB_NAME represents relationships between vocabularies used as specific data values (for objects in context models) and vocabularies in a natural language. For example, the symbol 'ALLHOS' means 'enemy' or 'hostile' in English, a relationship which can be represented by an instance of ATTRIB_NAME:

  ([attributeName-1] of ATTRIB_NAME
  (attribName forceCode)
  (attribValue ALLHOS)
  (alternatives "enemy" "hostile"))

When the word ‘enemy’ or ‘hostile’ appears in a free text sentence, CFTI can refer to it as the ‘ALLHOS’ value of the attribute ‘forceCode’ through use of a CLIPS rule (Figure 4-18).

```
(defrule find-NAME_VALUE-direct
   (mg  (order ?order)
        (body ?key)
    )
   (object (is-a ATTRIB_NAME)
        (attribName   ?attr_name)
        (attribValue  ?attr_value)
        (alternatives $? ?key $?)
    )
=>
   ;actions
)
```

A word from a text string

A PATTERN instance refers this word as an attribute value in a context model.

Figure 4-18: A CLIPS rule to search attribute values from a text string.

- WORD_TO_TOKEN represents relationships between TOKEN (i.e., an intermediate class representing common context knowledge, see 4.5.2.2) and vocabularies in a natural language. For example, the words ‘meters’ or ‘miles’ normally refer in English to the concept of distance, and the words ‘mph’ or ‘kph’ to the concept of speed:

  ([distance-1] of WORD_TO_TOKEN
  (tokenName DISTANCE)
  (alternatives "feet" "meters" "miles" "kilometer"))
  ([twoWordsSpeed-1] of WORD_TO_TOKEN
  (tokenName SPEED)
  (alternatives "mph" "kph"))

When these words appear in a free text sentence, the CFTI then is able to represent them as corresponding TOKEN objects through use of a CLIPS rule (Figure 4-19).

```
(defrule find-speed-token
   (mg (order ?order)
       (body ?num&:(numberp ?num))
       (group ?group))
   (mg (order ?order2&:(eq (+ ?order 1) ?order2))
       (body ?unit)
       (group ?group))

   (object (is-a WORD_TO_TOKEN)
        (tokenName ?token)
        (alternatives $? ?key $?))
   (test (eq ?key (str-cat ?unit)))
=>
   ;action
)
```

Two words appear next to each other in a text string, and the 1st word is a number.

The 2nd word refers to a TOKEN.

Figure 4-19: A CLIPS rule to search TOKEN from a text string.

```
(defrule word_convert
   (message ?mg)

   (object (is-a WORD_CONVERT)
        (code ?code)
        (alternatives $? ?key $?)
   )
   (test (str-index (str-cat " " ?key " ") ?mg))

=>
   ;action
)
```

Figure 4-20: A CLIPS rule to find a word that has a common form.

- WORD_CONVERT represents relationships between words and their common forms. For example, words 'two' and 'three' normally refer in English to the numbers '2' and '3':

  ([numWord-2] of WORD_CONVERT
  (code "2")
  (alternatives "two" "a pair"))
  ([numWord-3] of WORD_CONVERT
  (code "3")
  (alternatives "three"))

  When these words appear in free text sentences, CFTI attempts to convert them into their common forms through use of a CLIPS rule (Figure 4-20).

- PHRASE_CONVERT represents relationships between multi-word phrases and single words with the same meaning. For example, 'miles per hour' and 'mph' have the same meaning, and this relationship can be represented by an instance of PHRASE_CONVERT:

  ([phraseSpeed-1] of PHRASE_CONVERT
  (code "mph")
  (alternatives "miles per hour" "miles per hr"))

  When the phrase 'miles per hour' appears in a free text sentence, the CFTI attempts to convert it into 'mph' through use of a CLIPS rule (Figure 4-21).

```
(defrule phrase-convert
   (message ?mg)

   (object (is-a PHRASE_CONVERT)
        (code ?code)
        (alternatives $? ?key $?)
   )
   (test (str-index (str-cat " " ?key " ") ?mg))

=>
   ;action
)
```

Figure 4-21: A CLIPS rule to find a multi-word phrase that may be replaced by a single word.

4.5.2.2. TOKEN Classes

In addition to the representation of contextual knowledge, CFTI provides facilities for the representation of common context knowledge. Common context knowledge is normally independent of specific context. For example, the words 'mph' and 'kph' normally refer in English to the concept of speed, and the word 'northeast' normally refers to the concept of orientation. TOKEN classes are designed to represent common context knowledge in the CFTI (Figure 4-22).

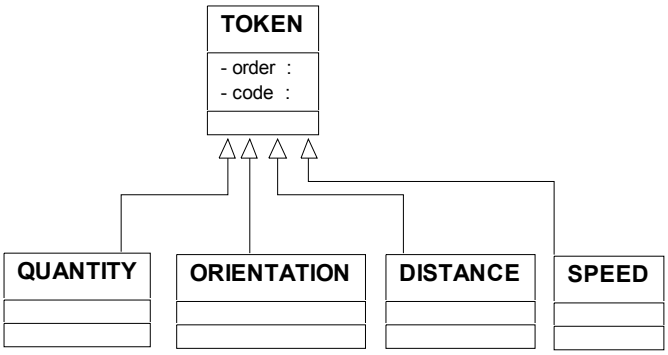


Figure 4-22: The TOKEN classes.

The descriptions and system uses of TOKEN classes are as follows:

- QUANTITY represents the magnitude of objects. For example, in the sentence 'there are 5 apples on the table', '5' represents the magnitude of the set of apples. While this type of knowledge may be readily understood by humans, an appropriate representation is necessary to achieve the desired interpretation by computer systems. The CFTI searches and represents information related to magnitude through the use of QUANTITY class objects (Figure 4-23).

```
(defrule find-Quantity
   ?mg1 <-(mg (order ?order)
              (body ?key&:(integerp ?key))
          )
=>
   (retract ?mg1)
   (make-instance of QUANTITY
      (order ?order)
    (code ?key)
   )
)
```

There is an integer in a free text sentence.

The found integer is represented by an instance of QUANTITY class.

Figure 4-23: A CLIPS rule to find an integer in a free text sentence.

- ORIENTATION represents the concept of direction. For example, the words 'northeast' and 'north-east' can refer to a geographic direction which is 45/360 compass degrees. This knowledge may be represented by an instance of the ORIENTATION class :

```
(orientation-2 of WORD_TO_TOKEN
  (tokenName ORIENTATION)
  (tokenValue 45)
```

(alternatives "north-east" "northeast" “north east”))

The CFTI can search and represent this information through the use of a CLIPS rule (Figure 4-24).

```
(defrule find-chunk
   ?mg1 <-(mg (order ?order)
              (body ?word)
          )
   (object (is-a WORD_TO_TOKEN)
        (tokenName  ?className)
        (tokenValue ?value)
        (alternatives $? ?key $?)
   )
   (test (eq ?key (str-cat ?word)))
=>
   (retract ?mg1)
   (make-instance of ?className
      (order ?order)
      (code ?value)
   )
)
```

A word in a free text sentence.

The word refers to a TOKEN

Instantiate a TOKEN object.

Figure 4-24: A CLIPS rule to instantiate a TOKEN object.

- DISTANCE represents the concept of length between two geographic locations. For example, ‘miles’ and ‘kilometers’ are units of distance measurement. This knowledge can be represented by an instance of DISTANCE:

  (distance-1 of WORD_TO_TOKEN
    (tokenName DISTANCE)
    (alternatives "meters" "miles" "kilometers"))

  The CFTI can represent this information through the use of a CLIPS rule (Figure 4-25) when these words appear in a free text sentence.

```
(defrule find-distance
   ?mg1 <-(mg (order ?order)
              (body ?num&:(numberp ?num))
              (group ?group)
          )
   ?mg2 <-(mg (order ?order2&:(eq (+ ?order 1) ?order2))
              (body ?unit)
              (group ?group)
          )

   (object (is-a WORD_TO_TOKEN)
        (tokenName DISTANCE)
        (alternatives $? ?key $?)
   )
   (test (eq ?key (str-cat ?unit)))
=>
   (retract ?mg1 ?mg2)
   (bind ?value (convertDistance ?num ?key))

   (make-instance of DISTANCE
      (order ?order)
      (code ?value)
   )
)
```

Two words appear in a free text sentence, and the 1st is a number.

The 2nd word refers to DISTANCE TOKEN.

Instantiate a DISTANCE object.

Figure 4-25: A CLIPS rule to instantiate a DISTANCE object.

- SPEED represents the concept of velocity. For example, ‘mph’ and ‘kph’ normally refer to the velocity of an object. (See Figure 4-19 for the usage of this class).

The TOKEN classes described above represent a small portion of common context knowledge. There exists a significant potential for future enhancements.

4.5.3. CFTI Object Model

In summary, the CFTI object model (Figure 4-26) contains three components: CHUNK (i.e., direct representation of a free text sentence), PATTERN (i.e., representation of contextual knowledge about words), and TOKEN (i.e., representation of common context knowledge about words).

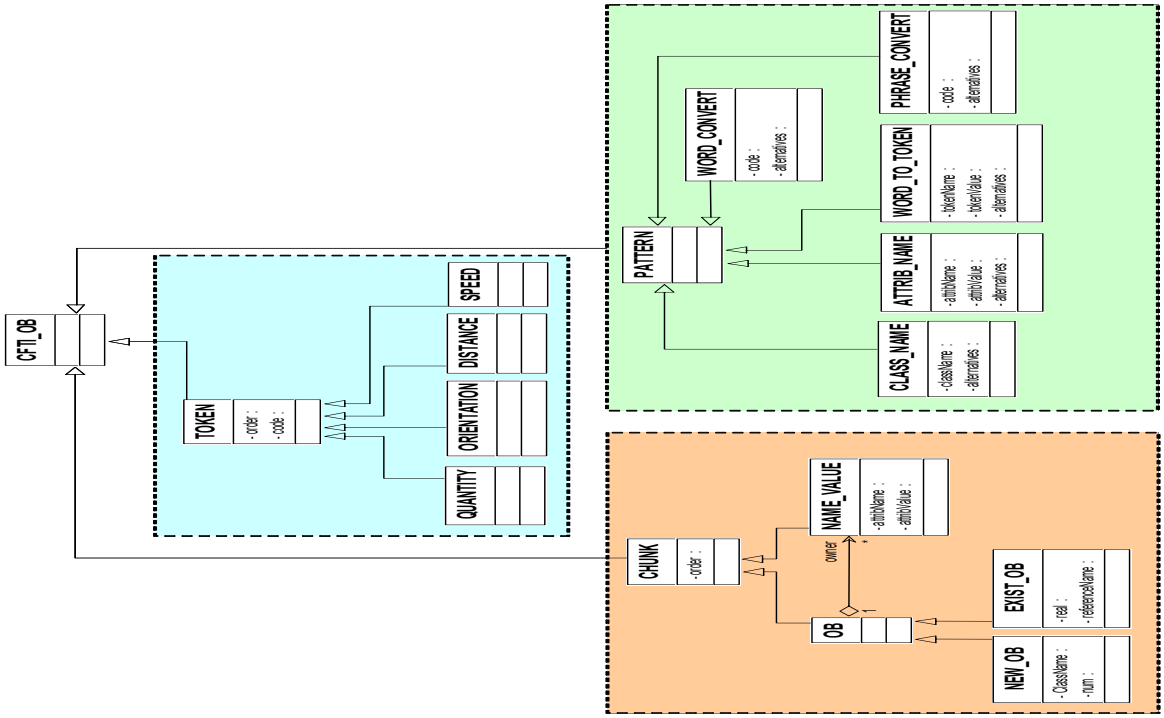


Figure 4-26: CFTI object model.

4.6. Testing Scenarios

In order to test CFTI, we developed a simple driver which obtains free text input from a CLIPS 6.20 console, interacts with the CFTI components (i.e., Link Grammar, Lisp simulator, and mapping engine), and represents the meaning of the sentence by manipulation of objects in a context model (i.e., Mini-IMMACCS).

4.6.1. Creation of New Objects

Understanding of free text is a process of model representation of the meaning inherent in the text. For example, the meaning of a sentence "three enemy tanks are 400 meters north at 56 miles per hour" may be represented by three Tank objects with corresponding attributes in the Mini-IMMACCS model (Figure 4-27).

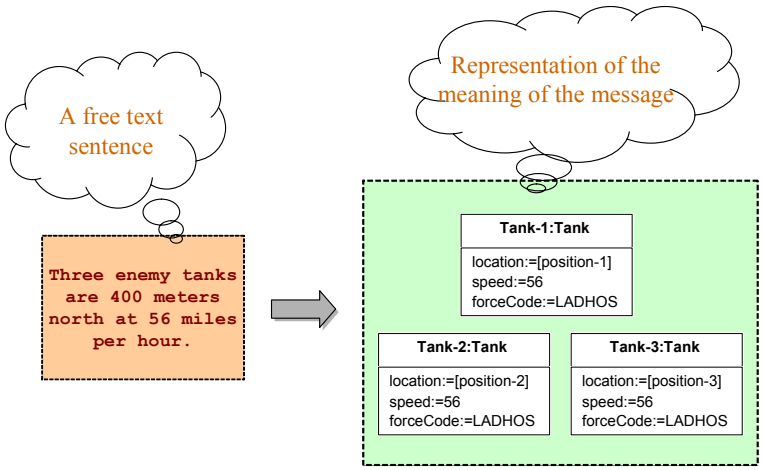


Figure 4-27: Creating new objects for meaning representation.

The following scenario demonstrates the process of running the above example in the CFTI system:

- Load and run 'mini.bat' in CLIPS 6.20, the console prompts for a free text input (Figure 4-28).

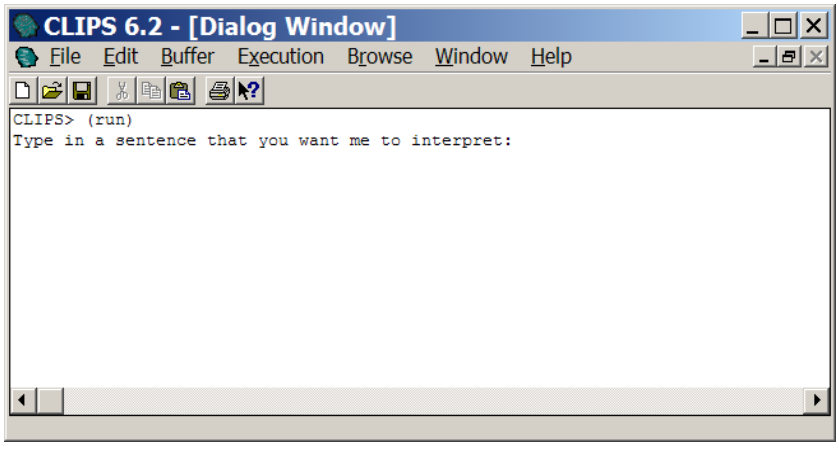


Figure 4-28: Screen shot of CLIPS 6.20 console after running 'mini.bat'.

- Type the sentence “Three enemy tanks are 400 meters north at 56 miles per hour”. The system will process this text string and create three Tank objects in the context of the Mini-IMMACCS model (Figure 2-29).

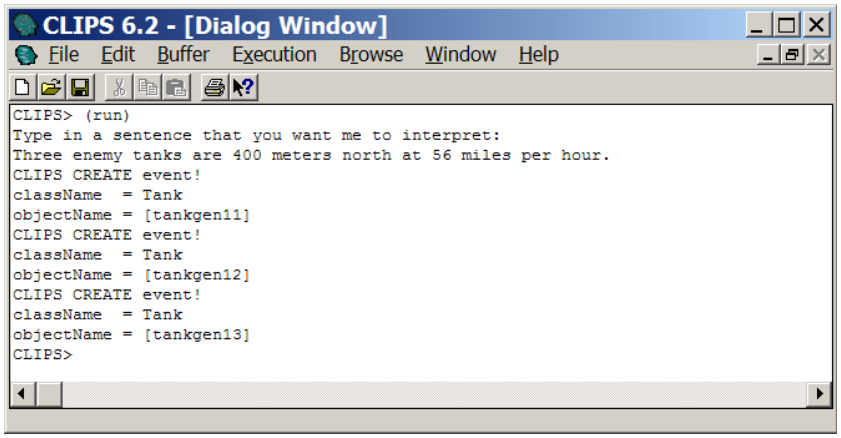


Figure 4-29: Creation of three Tank objects.

- By printing out the attributes of a newly created Tank object, it shows that the values are set correctly based on the original sentence. For example, ‘enemy’ is represented by setting the object’s ‘forceCode’ attribute to ‘LNDHOS’, and ’56 miles per hour’ is represented by setting the object’s ‘speed’ attribute to ‘56’ (Figure 2-30).

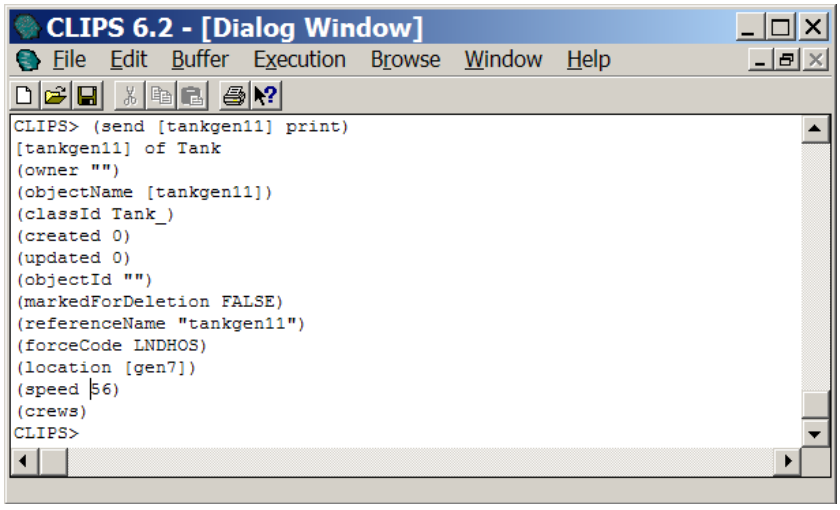


Figure 4-30: Attributes of a Tank object.

4.6.2. Update of an Attribute of an Existing Object

An object in a sentence may be an object which exists in a context model, or it may be a new object (Chapter 4.5.1). For example, if an object named ‘tank-1’ exists in the model, then the meaning of the sentence “tank-1’s speed is 46 mph” may be represented by setting / changing the ‘speed’ attribute of this object to ‘46’ (Figure 4-31).

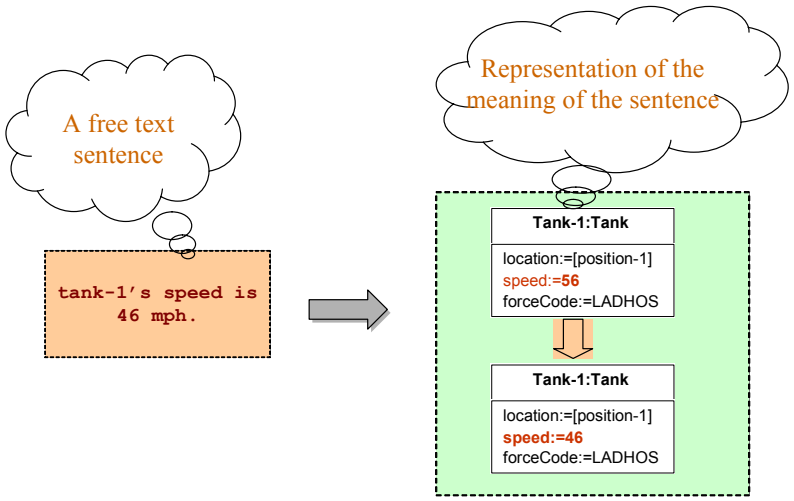


Figure 4-31: Update attributes of an object.

The following scenario demonstrates an actual run of the above example in the CFTI system:

- Continuing with the previous (Chapter 4.6.1) scenario, typing "tankgen11's speed is 46 mph" (Figure 4-32).

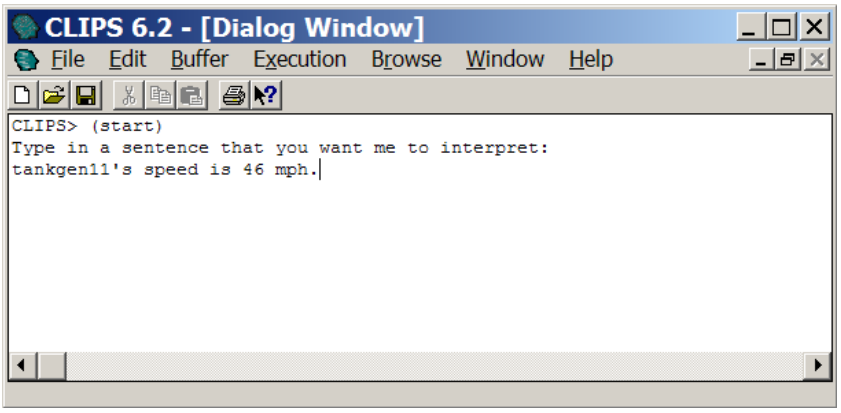


Figure 4-32: Input a sentence in CLIPS console.

- Printing, after the system quits running, of the speed of object 'tankgen11' with a value of '46' indicates that the CFTI has interpreted the meaning of the sentence correctly (Figure 4-33).

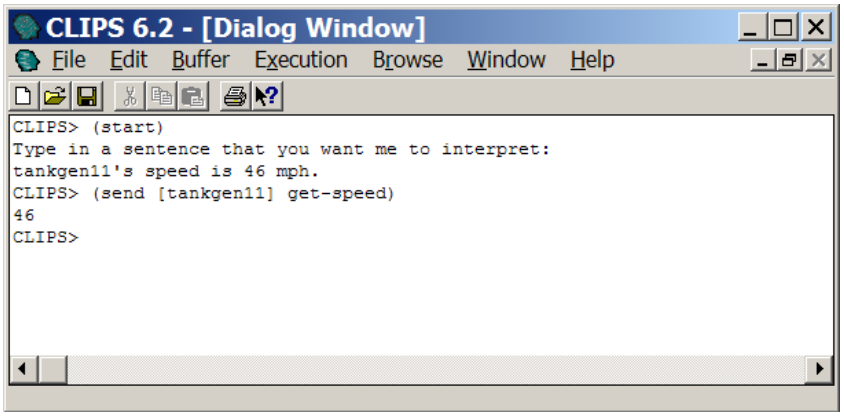


Figure 4-33: Value of an object update by CFTI.

4.7. Chapter Summary

The CFTI Implementation includes five components: Link Grammar, Lisp Simulator, WordNet database, a mapping engine, and a context model. The Link Grammar and Lisp Simulator process syntactic knowledge; the WordNet database provides lexical knowledge about words; the mapping engine consists of CLIPS rules which utilize the lexical and contextual knowledge for meaning retrieval ; and the context model provides: 1) contextual knowledge about words; and, 2) a representation of the meaning of a free text sentence.

The CFTI object model is composed of three sets of classes: CHUNK, PATTERN, and TOKEN. CHUNK classes are used as representation of free text sentences; PATTERN classes are used as representation of contextual knowledge; and TOKEN classes are used as representation of common context knowledge about words.

When the CFTI is tested against a small context model, Mini-IMMACCS, the system successfully represents the meaning of a free text sentence through manipulation of a context model. This is to say, the premise proposed by this project is correct.

5. Conclusions

Three objectives were stated in the introduction of this project:

- to research and identify generally essential components of NLU systems resulting in a theoretical product that leads to the development of CFTI;
- to research existing tools in the field of NLU, and to select and integrate one or more into CFTI if appropriate; and
- to develop the proposed CFTI system in order to demonstrate strategies of extraction and representation of information from free text sentences when a context model is provided.

Results of the first objective indicate that a NLU system needs to model the human understanding process. Humans utilize natural language to think about the real world, and ontologies are developed by humans as models of the world for use in computers. A process of understanding free text is a process of model representation of the meaning inherent in the text. A NLU system can only understand things that are representable in its context model. Direct and indirect mapping relationships exist among vocabularies used by ontologies and vocabularies used by natural languages. The capture and utilization of these relationships is key to the development of NLU systems. The quality of interpretation of free text is strongly dependent on the quality of the context models.

Studies of the second objective lead to the conclusion that two existing components, Link Grammars and WordNet, are among the most effective language tools, and were therefore selected and incorporated into the CFTI. It is necessary for NLU systems to be able to address syntactic and semantic aspects of natural language. CFTI utilizes Link Grammar to parse syntax, and WordNet to process lexical semantics.

Results of the third objective include CFTI design and implementation through use of CLIPS 6.20. The CFTI system contains five components: Link Grammar, Lisp Simulator, WordNet database, a mapping engine, and a context model. The Link Grammar and Lisp Simulator process syntactic knowledge; the WordNet provides lexical knowledge about words; the mapping engine is composed of CLIPS rules for meaning retrieval from free

text sentences; and the context model provides contextual knowledge about words and representation of meanings of free text sentences. While a context model is required by the system, change from one context model to another does not require significant system reconfiguration.

The resultant CFTI system demonstrates the capability of interpretation and representation of meanings of free text sentences based on a relatively small-sized context model (i.e., Mini-IMMACCS), which suggests promise for additional research.

Extension of the use of the WordNet database may constitute a potential additional research focus. The WordNet provides the CFTI with lexical knowledge about the English language; however, the current implementation includes only two types of relationships: synonymy and irregular forms of words. Inclusion of additional types of relationships may allow the WordNet to significantly enhance CFTI's understanding of natural language.

In CFTI, TOKEN classes are developed as an experiment for the representations of common context knowledge (e.g., speed, orientation, distance, and quantity). Because common context knowledge is not normally constrained by a specific context, it is reusable. Expansion of TOKEN classes may also enhance the capability of the system.

The CFTI system has been tested with a relatively small-sized context model. While an assumption that the system would perform similarly when tested with larger-sized models may be valid, conducting such tests constitutes another potential research focus, which could facilitate the abilities of the CFTI to be used in practical applications.

The CFTI currently assumes that a context model is static, which constrains the system from understanding subjects outside of the model. Future work into extensible ontologies and machine learning would significantly benefit the research study presented in this project.

Acknowledgements

This work was supported in part by the Collaborative Agent Research Center at California Polytechnic State University (Cal Poly), San Luis Obispo, California. We greatly appreciate the availability of the tools mentioned, in particular CLIPS, Link Grammar, and WordNet; without them, this work would have been practically impossible.

References

1. Allen, J. (1995). Natural Language Understanding. Redwood City, California: Benjamin/Cummings Publishing Company.
2. Ballim A., Pallotta V. (1999). Robust Parsing Techniques for Semantic Analysis of Natural Language Queries. VEXTAL'99, November 22-24 1999, Venice - Italy.
3. Baud R., Lovis C., Alpay L., Rassinoux A., Scherrer J., Nowlan A., Rector A. (1993). Modelling for Natural Language Understanding. In: Safran C (ed). Proceedings SCAMC 1993. New York: McGrawHill.
4. Chandrasekaran, B., Josephson, J., Benjamins, V. (1999). What Are Ontologies, and Why do We Need Them? IEEE Intelligent Systems, Vol. 14, No. 1 (January/February 1999), pp. 20-26.
5. Chauchard, P. (1964). Language and Thought. New York: Walker and Company.
6. Embley, D., Campbell, D., Smith, R., and Liddle, S. (1998). Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents. Proceedings of ACM

7th International Conference on Information and Knowledge Management – November 1998, Bethesda, MD.
7. Fellbaum, C. (1999). WordNet: An Electronic Lexical Database. Cambridge: MIT Press.
8. Gruber, T. (1993). "A translation approach to portable ontology specifications", in Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge-Based Systems. 5(2), June 1993.
9. Melcuk, I. (1988). Dependency Syntax: Theory and Practice. New York: State University of New York Press.
10. Miller, G. (1990). Wordnet: An Online Lexical Database. Int'l J. Lexicography, Vol. 3, No. 4, 1990, pp. 235-312.
11. Minsky, M. (1985). The Society of Mind. New York: Simon and Schuster.
12. NASA. (2002). CLIPS Basic Programming Guide, NASA JSC-25012.
13. Piaget, J. (1959). The Language and Thought of the Child. London: Routledge and Kegan Paul.
14. Sleator, D., and Temperley, D. (1991). Parsing English with a Link Grammar. Carnegie Mellon University Computer Science technical report CMU-CS-91-196.
15. Temperley, D., Sleator, D., and Lafferty, J. (1999). [online] An Introduction to the Link Grammar Parser. Technical report, March 1999. Available: http://www.link.cs.cmu.edu/link/
16. The American Heritage. (2000). Dictionary of the English Language, Fourth Edition. Boston: Houghton Mifflin Company.
17. Zaenen, A., and Uszkoreit, H. (1996). Language Analysis and Understanding. In Cole, R., Mariani, J., Uszkoreit, H., Zaenen, A., and Zue, V., [editors], Survey of the State of the Art in Human Language Technology. Cambridge: Cambridge University Press.