PLINK: An Intelligent Natural Language Parser

Christian Robert Huyck

1 August 1994

Contents

List of Figures

Chapter 1

Introduction

The past forty years have seen the development of a new approach to the study of language. The advent of the computer has enabled linguists, and more recently computational linguists, to formalize their models, test them, and actually process language. These natural language systems have been used in grammar checking, speech recognition, and information extraction. They will be needed to navigate the vast amount of information that the computer age has given us, but to do this effectively these programs must generate the correct interpretations of text more rapidly.

Early attempts at understanding natural language were centered on the form of grammar rules (formal languages). This work was productive but led to the distinction between competence and performance [10]. The grammars could generate interpretations for sentences that humans could not understand. The grammarians ignored this problem, noting that they were not concerned with human errors or memory limitations. Without these limitations, very complex formal languages were created to account for natural language, but the complexity of these languages made the implementation of algorithms that used them too expensive.

More recently, these human limitations have engendered more research. Beginning with Kimball [27], human performance has been studied more seriously [15], [12], [25] and [32].

This dissertation is concerned with making a computer generate interpretations of sentences. These interpretations should be similar to the interpretations a human would generate, and the method used to generate the interpretations should be similar to the method a human uses. In particular, it is centered around a natural language system which I have developed; it is composed of a knowledge base and a knowledge driven parser, and it attempts to parse in a human-like fashion.

There are three basic goals that my system was designed to achieve.

1. linear parsing[1]
2. generation of correct interpretations
3. extensibility

The first two goals, parsing in linear time and generating the correct interpretations, are directly linked to human language processing. People have difficulty interpreting some sentences that are known as garden path sentences. My system should fail on these sentences, but it should still fail in linear time. The third goal solves a software engineering problem; my system should be extensible.

It appears that humans interpret language in real time. They do not slow down as a sentence gets longer and they can read for a long time at the same rate. It would be advantageous for a natural language system to exhibit the same properties. If a system processes sentences in linear time, then it will not slow down while parsing longer sentences. This makes the parser faster, and since natural language systems will be used for 'real' tasks, they will need to be fast. Furthermore, algorithms that slow down on longer sentences are not psychologically plausible because people do not slow down.

[1] Linear time means that if a sentence is **n** words long, then the sentence is processed in roughly **Cn** time, where **C** is a constant.

The second goal is to generate the correct interpretation. An Natural Language Processing system (NLP) system should generate the solution that is similar to the one a human would generate. Any solution that is not similar, is not correct. In general, given a grammar, a sentence can have multiple interpretations which are supported by the grammar. That is, sentences are often ambiguous. Humans appear to omit most of the interpretations and settle on one or a few interpretations [14].

The third goal is more pragmatic; the system should be extensible. Building a natural language processing system is a very complex task which takes a very long time. If the system can be developed incrementally, shortcomings can be repaired. My system has been designed so that a significant portion of language can be initially processed, but new portions of the language can be added later without significantly affecting the old portions.

These goals are significant, but in the process of developing this system, another problem presented itself. Natural language systems are very complex and can be easily made to account for a large number of sentences. However, when new sentences are presented, the systems often break down. Testing should attempt to uncover these break downs, so systems should be tested on text that they have not seen. Moreover, these sentences should be the type of sentences people normally generate. They should cover a broad set of semantic and syntactic phenomena and they should be designed for humans not for natural language systems. That is, they should be selected from an open domain.

My thesis is that intelligent parsing can improve the efficiency of natural language processing, and it is necessary to correctly model human language processing. The system I have implemented functions both linearly and rapidly and generates correct interpretations. This system is easily modifiable, has been tested on a wide range of syntactic and semantic phenomena including sentences that it had not previously seen, and it has performed well in tests in an open domain.

1.1 Overview

The approach that I have taken is to generate a complete syntactic and semantic representation; however, I would like to generate the representations that a human would generate. Of course it is very difficult to say which representation a human generates (see section 1.4.3), but some representations can be said to be those that humans create [15]. This makes psychological validity very important. Other subfields of Artificial Intelligence (AI) are less concerned with psychological validity; if they can get a better answer in some clever non-human way then so be it. This soft AI answer does not work for NLP; there is no better answer. The correct answer is the answer a human generates! So if the NL system sees

Example 1.1.1 *Time flies like an arrow.*

it should the answer

Example 1.1.2 *Time goes quickly.*

rather than the answer

Example 1.1.3 *Creatures called* Time Flies *enjoy an arrow.*

This approach to natural language processing is very human-centered. If a system generates the interpretations humans do, it has a case for being a good psychological model. However, some systems, whose goal is to generate the human interpretation, function in a fashion that is definitely not the way humans function. Ephratt [14] has developed a system which generates all of the possible parses, and then evaluates each to find which is the best (i.e. the one a human would select). This is not how humans process text [26]; still the information Ephratt's system used must be related to the information a human uses. A better psychological model would generate the correct interpretations at each step of processing. Other factors are relevant, but this step by step performance metric is very potent. If, at each step, a system is creating representations that are similar to those a human creates, the system is probably functioning to some degree like humans.

In addition to this goal of generating a complete representation, natural language systems should obey certain constraints. One constraint is to generate the representation in linear time as humans appear to.

If a system generates the interpretation in linear time, it has an even stronger case for being a valid psychological model. Moreover, linearity is useful in real systems because it assures they will be fast. Thus,

such a system will have something to say about the information a human uses while processing sentences, and it will be a useful program in its own right.

Up to this point, I have tacitly assumed the existence of a computational system. It is useful to develop an informal model. However, a formal model implemented as a piece of software is better. This software must be rigorously implemented so it cannot be entirely ad-hoc, it can be rigorously tested on real data (eg. text), and it can be used in applications that need to process natural language.

Developing a natural language system is very complex. It would be useful if such a system could be developed incrementally so that it could be tested on various parts of the language. Similarly, human language develops; 3-year olds do not understand language as well as adults. For both of these reasons, it is good for a system to be easily modifiable. A system that can learn about new linguistic phenomena would be even more useful. While my system might be used in a learning system [21], it does not currently learn.

I have developed a system, PLINK, which uses Unification-Based Grammar [45] as a declarative knowledge formalism. It is a modification of an earlier system called LINK [34]. The majority of the new work for this system has been on a new parsing mechanism that is a knowledge driven, stack-based mechanism. This mechanism also allows incremental development of the knowledge base.

1.2 The Problem

If it is so desirable to generate complete syntactic and semantic representations, then why do so few systems generate them? The reason is that it is very difficult to generate complete representations. Natural languages are highly ambiguous. Example 1.2.1 has 156 possible syntactic interpretations.

Example 1.2.1 *The cars raced by the spectators crowded into the stands at over 200 mph on the track at Indy.*

Raced and *crowded* can be read as the beginning of relative clauses or as the main verb, and the prepositional phrases can be attached to the verbs or to earlier noun or prepositional phrases.

Simple algorithmic approaches must generate all of these interpretations. In general, the number of representations that a sentence may have is on the order of $\mathbf{2}^n$, where $\mathbf{n}$ is the length of the sentence[2]. As sentences get longer, [3] even the most efficient simple algorithmic methods must fail to process text in a reasonable time.

Another problem, related to speed due to ambiguity, is resolving the ambiguity. When a sentence has 156 representations, how is the correct representation selected? Consider the sentence:

Example 1.2.2 *Pat went to the store for bread.*

It has at least two possible interpretations. The first interpretation is that *for bread* modifies *went*; so, to get bread is the reason that Pat went to the store. The second is that *for bread* modifies *store*; so, the store is a bread store. The first interpretation is the correct one, but in the sentence

Example 1.2.3 *Pat went to the school for the blind.*

the correct interpretation is parallel to the *bread store* interpretation.

Ambiguity is the major problem that PLINK addresses. However, as in most AI problems we have a good example for the correct solution: humans.

How do humans select the correct interpretation and process in real time? If it is assumed that humans apply something like context free grammar rules, then they could solve the problem by keeping only a small number of possibilities active. Indeed this seems to be the case [26]. Humans attempt to determine the status of each word in the sentence and in the discourse before they go on to the next word. If they use

[2] An example is the grammar consisting of the rule $\mathbf{A} \rightarrow \mathbf{AA}$, where A is both a terminal and a non-terminal. Any time a new A is added, it creates at least twice as many possible interpretations. If there are J interpretations for a string of length n, then there are at least $2J$ interpretations for a string of length $n+1$. There are J interpretations for the first A combined with all J combinations of the last n As, and J interpretations for the last A combined with all J combinations of the first n As. Ignoring interior combinations each additional A doubles the number of interpretations that can be generated. Since a string of length 2 has 1 interpretation, a string of length n will have on the order of $\mathbf{2}^n$ interpretations.

[3] They can get very long; for example Moby Dick contains a sentence that is 469 words long. [36]

grammar rules, they must choose which of the several possible grammar rules to apply and ignore the rest. That is, humans resolve ambiguity locally (see section 2.5.1). The kinds of mistakes that humans make show that they resolve ambiguity locally; incorrect local decisions lead to garden pathing problems (see section 5.5). In this manner they can parse linearly, and generate the correct solution.

1.3 Grammars and Parsing

Chomsky [10] says that linguistic competence relates to

> an ideal speaker-listener, in a completely homogeneous speech-community, who knows its language perfectly and is uneffected by such grammatically irrelevant conditions as memory limitations, distractions, shifts of attention and interest, and errors (random or characteristic) in applying his knowledge of the language in actual performance. (pg. 3)

Until recently this idea has dominated thought about computational linguistics. The focus of Chomsky's work is on grammatical formalisms. The question is what type of formalism (formal language) effectively represents natural language. A grammar that effectively represents natural language is a competence grammar.

Chomsky defines a hierarchy of grammars [9]. The least powerful grammar is the regular grammar, which can be interpreted by a finite state automaton. There is an argument for language being represented by regular grammars. Humans are finite, so theoretically, there is a finite automaton which can simulate a human being; however, the number of states that would be needed would be enormous. Regular grammars can also be easily parsed in linear time. However, certain natural language constructs cannot be easily represented by regular grammars. While there is some disagreement (see section 2.2.1), a language that is at least as complex as context free grammars (CFG) is generally thought to be necessary to describe natural language.

Unfortunately, general CFGs cannot be parsed in linear time. Moreover, a string of length **n** can have on the order of $\mathbf{2}^n$ different interpretations over a particular CFG. Still, there are structures that occur in natural language that cannot be represented by a CFG. An example is the *respectively* construct (e.g. Pat and Chris like Hondas and Fords respectively.). This is an example of the formal language $L = \{xx \mid x \in \{a \vee b\}*\}$

Here we see the problem. A powerful grammar is needed to interpret natural language, but there is no simple algorithm that enables sentences to be processed using this grammar in a reasonable time.

A tacit assumption in this is that grammar rules are similar to structures in the brain (see [31]), and these structures are used in processing natural language. It is unlikely that simple context free grammar rules like

Example 1.3.1 *Sentence* → *Subject-NP Head-VP*

exist in the brain, but it is likely that something like a grammar rule exists at some level in the brain. Rules probably exist in the cell assembly level[4] These , that is at the level Newell calls the neural circuit [40].

This assumption is important because these rules must somehow be activated, and once activated they are applied. The grammar rules in my system correspond to the grammar rules in the brain. The information that is used to activate the rules in the brain should correspond to the information that PLINK uses to select its grammar rules. It is not necessary to show the exact form of the grammar rules in the brain. Instead, whatever mechanisms humans use for parsing are probably guided by the same sort of information that PLINK uses to guide its parsing.

Grammatical rules imply a notion of linguistic competence. If a sentence matches the grammar, then it is grammatical. However, the idea of a competence grammar has a problem: it implies that some sentences that people do not understand are grammatical. The now infamous garden path sentence

Example 1.3.2 *The horse raced past the barn fell.*

is a sentence which is grammatical but is not understood by most people. The sentence can be made understandable by adding a relative clause marker:

[4] A cell assembly is a group of neurons that are interconnected so that they are mutually exciting. This assembly can have persistant activation even when individual neurons are fatigued. Thus a concept, in this case a grammatical rule, is stored as an assebly of neural cells [20].

Example 1.3.3 *The horse that raced past the barn fell.*

The idea of a sentence that is grammatical but cannot be understood is a strange idea, but the idea is made necessary by a competence grammar. If we accept competence grammars, then we must raise another question: *what sentences can be understood?* This aspect of performance has recently been the topic of much study (see [12] and [35]).

While Chomsky ignores many human factors, this new question makes human factors much more important. With a competence grammar the important question is *what is the grammar*. When performance is considered, grammar is still important; however, the question *which grammar rules are applied* is now much more important.

When grammatical rules are applied to text to generate a representation of a sentence, it is called parsing. As has already been stated, for any reasonable formal language, there is no simple linear algorithm to correctly parse text. Therefore a parser must use more knowledge.

My solution is based on this observation:

to eliminate global ambiguity, eliminate local ambiguity.

This sounds simple, but it is quite complex. A great deal of information needs to be present to make the choice between several rules. Syntactic, semantic, and contextual information can all affect the choice of which rule to apply.

Syntactic information is very important. In English, syntax is essential to determining the correct representation of a sentence. In the fragment

Example 1.3.4 *the dogs*

a rule can be applied to the plural noun *dogs* to make a noun phrase; however, syntactic context suggests that *the dogs* is the correct noun phrase to create.

Semantics is also very useful for rule selection.

Example 1.3.5 *Pat hit the nail with the hammer.*

In the example 1.3.5 semantics suggests that the hammer should be the instrument of the verb *hit*, and not a modifier of *the nail*.

Example 1.3.6 *Pat hit the nail with the red head.*

In example 1.3.6, *red head* modifies *the nail*.

Context can also affect which rule is selected. In the sentence

Example 1.3.7 *I ate the cake with the clown.*

The standard interpretation has *the clown* modifying the cake; so there is a picture of a clown on the cake. However, if the previous sentence is

Example 1.3.8 *Bozo the clown brought me a birthday cake.*

then *the clown* suddenly becomes the coagent.

Syntax, semantics, and context are all used to make parsing decisions [12] [19]. Parsing relates to the activation of grammar rules, and in the human sentence processor parsing relates to the activation of grammar rules in the brain. So, these rules are activated by a combination of syntactic, semantic, and contextual knowledge.

Local information can be used to eliminate local ambiguity, which in turn will eliminate global ambiguity. My system uses local information to guess which local interpretation is correct. However, the system can guess incorrectly. I have assumed that this local guessing is similar to what humans do. This would account for garden path sentences like 1.3.3. A guess is made that *raced* is the head verb, but this guess is shown to be wrong when *fell* is read. My parser will occasionally fail, but it should fail when humans fail.

1.4 The Solution

My solution was to develop a computational system. To do this I selected a grammatical formalism and developed a parsing mechanism. Once the system was developed, it was tested to see how well it solved the problem. Did it parse, resolving ambiguity locally, and generate the correct interpretation? Did it fail when humans fail?

1.4.1 The Grammar

The goal I have declared is for PLINK to process sentences like humans. I have talked about grammars and I will use a Unification-Based Grammar, which is slightly more powerful than a CFG.

The use of syntax, semantics, and context in rule selection is relevant to the type of grammar used. In my system, I have chosen to use a Unification-Based Grammar (UBG) [45] [34]. It is a useful formalism for simultaneously generating a syntactic and semantic representation. It also enables syntactic, semantic, and contextual knowledge to be encoded in a small amount of computer memory.

PLINK simultaneously develops both a syntactic and semantic representation. If the system is to be psychologically plausible, it cannot wait until syntax is completed before generating a semantic representation. While the exact nature of the interaction between syntax and semantics is currently unclear, some semantic processing must be done before the end of the sentence. After all, when reading this sentence you have developed a semantic representation before you see the end of the sentence. Also if the system is using syntax and semantics to choose rules, both syntax and semantics need to be present.

Context free grammar rules are able to encode all of these types of information, but they can become quite unwieldy. For instance, assume there are two types of verb phrases, one that takes only humans as actors, and another that takes only dogs. A UBG can represent them in the same fashion and thus needs only one verb phrase grammar rule. A CFG needs two. As these differences mount, the size of the CFG grows very rapidly. The ability of UBGs to keep the size of the grammar reasonably small and use syntactic and semantic information was a major reason for its use in PLINK. UBGs are more thoroughly explained in section 2.1.1.

1.4.2 The Parser

The problem I am addressing is that NLP systems do not parse like humans. The obvious solution is to build a system that does parse like humans. How does one build such a system?

The first problem, as in most software problems, is to decide the input/output characteristics of the system. Input to the system is a single sentence. [5] The parser uses the words in the sentence to select grammar rules to apply. The application of these rules builds a complete interpretation of the the sentence. This interpretation is the output.

Processing one sentence at a time may seem simplistic at first. It may seem like too little information; after all, humans have surrounding sentences as context, intonational cues, and the local environment. A system that intelligently handles these other types of information in addition to the stand-alone sentence would be more powerful. But humans are able to process single sentences without additional context.

Intonational cues are unnecessary; humans can read. Environmental context is unnecessary; humans can read about the desert in a forest. Surrounding sentences are unnecessary; humans can read a single sentence. [6] Since the other types of information are unnecessary, and it simplifies the task, my system receives only one sentence at a time. [7]

What does it mean to receive a sentence as input? Many systems receive each word and punctuation marker in a sentence at once. This is not what humans receive. A person is unable to remember all the

[5] A grammar could be developed to parse more than one sentence. Such a system could make more effective use of context (see section 4.4). However, to limit complexity, PLINK will handle only one sentence at a time.

[6] Anaphor resolution is crucially linked to multiple sentence processing. Some context mechanism is obviously essential to human sentence processing and human sentence parsing (see section 4.4).

[7] PLINK, can be used to mark the end of sentences (see section 5.4.5). So, multiple sentences could easily be handled at one time. A minor adjustment would be needed to handle an unlimited number of sentences, though a context mechanism (see section 4.4) would be needed to handle an unlimited number of sentences correctly. This leads to many discourse and memory issues about which PLINK has little to say.

words in a fifty word sentence. Instead, humans receive one word or punctuation mark at a time[26]. Thus, the system should receive only one word at a time.

In addition to each new word, the system must have a representation of what has already occurred in the sentence. The representation used in PLINK is a stack. (See section 3.5 for further discussion on this topic.) The stack mechanism can be abused by allowing the creation of two types of unreasonable solutions: too many representations, and two-pass parsing. These are abuses because humans do not process in this fashion [26].

It is possible to use the stack to generate multiple, and in fact all, solutions. This can be done by applying all rules and using the stack to keep track of the results. In PLINK, the stack is limited to only one syntactic structure over a string of words. For instance, the string of words *The cat in the hat* can be represented by the stack of syntactic structures *determiner noun preposition determiner noun*, the stack *noun-phrase prepositional-phrase*, or the stack *noun-phrase*. Since there can be only one syntactic structure for a given string, the problem of multiple interpretations is virtually eliminated.

Another problem is that a stack can be used for two-pass parsing. Structure is built going left-to-right through a sentence, and then recombined going right-to-left. PLINK does not prohibit this because is uses right-to-left combination to handle right attachment problems (see section 3.5 for further discussion.) However, PLINK only functions in a right to left manner to account for right attachment.

Of course, humans do not have just the sentence as input. Humans also have a sophisticated grammar, a huge lexicon of words, and a lot of world knowledge. A limited version of this information is also provided to the system.

The parser functions by selecting a grammar rule and applying it. It repeats this process until the sentence is completely parsed, or until no rules are selected. The key to the parser then, is the rule selection mechanism.

The output of the rule selection mechanism is an ordered list of rules. The first rule is applied. If it fails [8], the second rule is applied, and so on, until a rule succeeds. When a rule succeeds, it modifies the stack, and the rule selection mechanism is called again. If an end state is reached, parsing is completed. An end state is reached when an end symbol is the only element on the stack and there is no more input.

As stated earlier, a complete syntactic and semantic representation is returned from parsing. Evidence suggests that humans do not generate such a representation because they cannot reproduce the syntactic structure of sentences. The result of human parsing can be approximated by just the semantic representation. However, it is useful to have the complete syntactic representation to determine if the system has generated the correct parse.

Now that we know what goes into the system and what comes out of the system, what happens in the middle? To a large degree, that is what the rest of this thesis is about. The main idea is that there must be a principled way of selecting rules using the information that is on hand. I have tried both raw LISP code and a rule based system for this principled mechanism. Both use syntax and semantics, and both have knowledge of the grammar rules. Both need to have all of this knowledge to correctly select the next rule. I code this knowledge in principled rule selection heuristics. Principled rule selection heuristics lead to principled natural language parsing.

1.4.3 Testing

I would like PLINK to parse sentences in a human-like fashion. How do I show that it parses in this manner?

The parser should select the correct, or human, interpretation. Since a sentence can have multiple interpretations, how can it be determined which interpretation is correct? Ford, Bresnan and Kaplan [15] used a group of subjects to find the correct interpretations of sentences. Unfortunately, this is a time consuming way of verifying interpretations. Except on Ford, Bresnan and Kaplan's sentence, I have been the judge of correct interpretations of the test sentences that I have used.

Of course, the correct interpretation depends on the actual grammar that is used. There is not agreement on which grammar correctly represents English. Therefore, I have attempted to use standard grammars in my systems that would not be controversial.

[8] The application of a grammar rule fails if the constituents do not unify with the grammar rule and with each other (see section 3.1.2).

In later version of PLINK, I was also concerned with correct partial interpretations. Is PLINK generating the correct interpretation along the way? This has been difficult to do because of the right attachment problem (see sections 2.3.1 and 3.5). However, by keeping the size of the stack small and resolving ambiguity locally, I have attempted to keep correct partial interpretations.

Natural language is infinite, so systems can be easily developed that process an infinite number of sentences. However, one difficult test of a natural language processing system is how broad a range of sentences it can process. A system should therefore be tested on a broad range or sentences. Since humans can read a broad range of sentences, and the goal of the system is to read like humans do, it should be tested on sentences selected from a corpus of sentences that were meant for humans. That is, it should be tested on sentences from an open domain. A PLINK knowledge base (KB) was developed on sentences from the microelectronics domain and 87 percent of the sentences were completely parsed. This was the domain used in the Fifth Message Understanding Conference [2], and was composed of newswire stories about chip fabracation. PLINK with the same KB was also tested on unseen sentences from this domain and it parsed 68 percent of them completely. The same KB was tested on sentences from another domain, Central American Terrorism, and parsed 58 percent of these sentences. This domain was used in the Fourth Message Understanding Conference [1], and was composed of newswire stories about terrorist events.

Finally, the human sentence processor occasionally fails. One type of failure is garden-pathing. I have tested my system on garden path sentences, and it fails on the sentences on which humans garden path and succeeds on non-garden path sentences.

1.5 Outline

The rest of this dissertation progresses as follows. First, I review some related works. This review should place my work in the natural language literature, and give some background. The third chapter is a description of the basic parts of PLINK, including a simple LISP-based parsing mechanism, and the results of this system on a small set of sentences. The fourth chapter describes my attempt to expand this initial system into an open domain of sentences. This expansion led to some problems that are first solved by an extension of the LISP-based parsing mechanism. However, the LISP-based mechanism is not easily extensible, so I also describe a more robust rule based mechanism. The fifth chapter presents the results of PLINK with this rule based mechanism, evaluates its performance on certain problems, and shows how it could be easily improved to include backtracking. I finish with some concluding remarks and some thoughts about future work.

Chapter 2

Related Work

Natural language systems have been developed for quite some time. In this chapter I will present some work by others in this field. It is by no means comprehensive, but it addresses five of the most relevant topics: declarative formalisms or grammars for language representation, simple parsing algorithms, the importance of parsing, full fledged parsing systems, and finally psychological evidence.

The idea of each of these sections is to more fully explain design decisions made in PLINK with regards to these topics. In section 2.1 I explain various formalisms including the Unification-Based Grammars which I use in PLINK. In section 2.2 I describe some simple algorithms for parsing these formalisms; these are insufficient for the task of human-like parsing because they cannot parse in linear time. Section 2.3 describes some human-like characteristics that a parser should exhibit. Section 2.4 describes some systems that attempt to do full-fledged natural language parsing. Finally, section 2.5 presents some psychological evidence that supports the approach to parsing used by PLINK.

2.1 Grammars

Early work on formal languages was done by Chomsky [9], in which he introduced context free grammars. Backus adapted context-free grammars into BNF notation, which was used for formally describing Algol 60 [39]. Efficient algorithms were designed to parse both regular languages in linear time, significant subsets of context-free languages in linear time, and all context-free languages in cubic time.

However, natural language is not easily understood in terms of formal language. The position of natural languages on the scale of complexity of formal languages is under debate, though most current systems assume that natural language is at least context-free. A key question for computational linguists is:

> Which formal language should be used to model natural language?

2.1.1 Shieber

Since Chomsky, there has been more work in formal languages. One of the most popular representations for natural language systems is the Unification-Based Grammar. Shieber [45] gives an excellent overview of the different variations of Unification-Based Grammars (UBGs). UBGs are Turing powerful but they can also represent information concisely.

UBGs are based on features and their values, also known as feature value pairs. I will denote features by lower case words, and values will start with capital letters. Examples of feature value pairs are *(number Singular)*, *(category Verb)*, and *(semantics Animate)*. A feature can have more than one part; *(agreement number)* is a valid feature so *((agreement number) Singular)* is a valid feature value pair. Structures are composed of several feature value pairs.

Example 2.1.1

(category) NP
(agreement number) Singular
(agreement person) Third

refers to a third person singular noun phrase.

Grammar rules can be defined using a UBG structure.

Example 2.1.2

$X_0 \rightarrow X_1\ X_2$
$< X_0\ category > = Sentence$
$< X_1\ category > = NP$
$< X_2\ category > = VP$
$< X_0\ head > = < X_2\ head >$
$< X_0\ head\ subject > = < X_1\ head >$

This is the (S $\rightarrow$ NP VP) rule which assigns the head of the NP to the *(head subject)* feature and the head of the VP to the *(head)* feature.

Unification is the operation that combines two, or more, structures. The structures are overlayed so that distinct features are just combined, but if the features from both structures overlap the values must also match. Structures cannot be unified if they have contradictory feature value pairs.

Example 2.1.3

category NP
(head type) Common
(head) Dog

The structure in example 2.1.3 might be used to represent the string *the dog*.

Example 2.1.4

category VP
(head type) Past
(head) Run

The structure in example 2.1.4 might be used to represent the string *ran*. Using the grammar rule of 2.1.2 and the verb structure of example 2.1.4 and the noun structure of example 2.1.3, the noun phrase is assigned to X_1 and the verb phrase is assigned to X_2. Unification combines the structures giving

Example 2.1.5

category Sentence
(head) Run
(head type) Past
(head subject) Dog
(head subject type) Common

One key aspect of UBGs is that multiple features can share the same value. By adding example 2.1.6 to the grammar of example 2.1.2, the rule guarantees that the result will have number and person agreement.

Example 2.1.6

$$< X_1\ agreement > = < X_2\ agreement >$$

2.1.2 Jurafsky

Jurafsky introduces Construction-Based Interpretive Grammars (CIG) in his 1992 thesis [25]. This is a member of a family of grammars known as Construction Based Grammars [30]. Jurafsky implies that CIG is more uniform than Unification-Based Grammars. That is, all linguistic information can be represented by CIG so it is uniform. While CIG is no more powerful than UBGs, it does solve, or at least ease, the right association problem (see section 2.3.1) by using cursors.

The only difference between CIG and UBG is that CIG introduces the cursor into the grammar. [1] The other structures of CIG correspond directly to UBG structures.

The cursor is used solely during interpretation. Structures are built, and they may be modified by future parts of the sentence. In a UBG new information can be attached anywhere in the existing structure, but in a CIG this new information may only be attached at the cursor. That is, the cursor marks the possible attachment site for future portions of the sentence. Consider the sentence of example 2.1.7:

Example 2.1.7 *Pat wanted the shirt.*

Figure 2.1: Cursor position

After *wanted* is read a structure like figure 2.1 is built. The position of the cursor indicates where *the shirt* will be attached.

Jurafsky states that all linguistic knowledge can be represented by CIG. I feel that this claim is too strong and would be more cogent if it merely stated that all declarative knowledge can be represented by CIG; however, even the weakened claim is a strong one. He shows that semantic, syntactic, and grammatical knowledge can be represented, and states that contextual, phonological and pragmatic knowledge can be represented. However, as will be shown below, UBGs can represent context, syntactic, semantic and grammatical information. Like Jurafsky, I will omit phonological and pragmatic information, but again UBGs are general enough to account for this type of information [21].

Lakoff [30] notes that there is not a clear division between different types of information.

> ...there is more likely a continuum between the grammar and the lexicon.

This is a claim that Jurafsky is trying to exploit. He attempts to treat different types of information in the same fashion. By representing information uniformly, and manipulating it uniformly, he makes them the same.

The cursor enables Jurafsky to solve the right association problem. A principle of language that Kimball [27] states is the principle of right association; terminal symbols optimally associate to the lowest non-terminal node. Most parsing algorithms prevent attachment of a phrase until it is completed. When this type of attachment is combined with the right association principle, a large number of phrases must often be kept. An example of the problem is

Example 2.1.8 *Pat wanted the shirt on the rack.*

In most other formal languages, *the shirt* cannot be attached to *wanted* until *on the rack* is attached to *the shirt*. In multiply embedded prepositional phrases a large number of phrases must be stored. Jurafsky avoids this by attaching the phrase early but allowing it to be modified (have something attached to it) later.

Unfortunately, Jurafsky does not extensively discuss the use of the cursor. It appears that it is introduced only from grammatical rules. Again unfortunately, this seems to weaken Jurafsky's claim of uniformity. Another problem is multiple cursors. For the above sentence, *on the rack* could also be attached to *wanted*, so there should be a cursor in its location slot. Jurafsky handles this by creating multiple interpretations. This limited parallelism is a side-effect of Jurafsky's interpretation (parsing) mechanism, which is more fully discussed in section 2.4.3.

2.1.3 $\overline{X}$ theory

The study of grammars involves more than the question of which type of formal language best describes natural languages. It also involves the form of the actual grammar; if one type of formal grammar is chosen, say a CFG, then a particular grammar, an instance of a grammar, needs to be built. Which instance of a CFG should be used, or what properties should such a CFG hold?

One set of properties is defined by $\overline{X}$ theory [11]. Chomsky summarizes the main point:

> a good case can be made that the lexical categories Noun, Adjective and Verb can appear in base forms with complements to form noun phrases, adjective phrases, and verb phrases.

[1] Jurafsky also uses frequency information, but that could be easily added to a UBG.

Figure 2.2: The Representation of *the dog with the bone*

The general idea is that the head noun has complements added to become a simple noun phrase. *Dog* can add *the big* to become the simple noun phrase *the big dog*. Similarly the verb *talked* becomes the simple verb phrase *would have talked*. I have used N and V and NP and VP. If the variable X is substituted for N and V we have XP, but P has been overriden by bar. Thus we have $\overline{X}$ theory.

2.2 Parsing Algorithms

The question of which grammar best represents natural language is linked to the algorithms for interpreting sentences into structures of these grammars. Attempts by researchers such as Tomita [48] to develop simple algorithms based on exhaustive search have met with some success, but their failings point out the problems of a this approach to natural language parsing.

2.2.1 Blank

One key thing to be noted about Regular Grammars is that sentences can be interpreted unambiguously into a Regular Grammar structure in linear time. Furthermore, they are traditionally modeled by Finite State Automata. Since all humans are finite, Blank argues that it must be possible to model human linguistic abilities with a Regular Grammar. While this is theoretically true, the size of the automaton that would be needed to model a human would be enormous.

Blank's [6] system has linearity in mind, and it uses a regular grammar with registers to account for embedded constructs. This approach has certain limitations since in English some constructs can be indefinitely embedded. The use of a finite number of registers means that any sentence that exceeds that number of registers cannot be interpreted.

Furthermore, the number of rewrite rules needed for any significant subset of English would be enormous. For instance, prepositional phrases can modify prepositional phrases and this modification can be recursive. One set of regular rules would be needed for each level of recursion. One CFG rule can account for this phenomenon. An infinite number of Regular rules would be needed, so Blank must set an arbitrary limit to the depth of the recursion.

Blank's selection mechanism is embedded into his parsing mechanism, so he really has little to say about rule selection. His work does, however, provide an excellent statement on linearity. Blank's results are impressive, but again his system only works on a small subset of English. It is entirely unclear that his system can work on a large subset of English, because his tests are on extremely simple sentences. He uses registers to avoid context free structure, but it is likely that he will need a very large number of registers for full English. As he uses more registers, his grammar must get much more complex. This complexity could easily overwhelm system development.

2.2.2 Chart and Active Chart Parsing

One way to speed parsing of CFGs is to use chart techniques. These techniques are centered around well-formed substring tables. Active chart parsing is an extension of chart parsing.

Well-formed substring tables are used to keep track of what has already been done. If a phrase like *the dog with the bone* has been processed, a tree like figure 2.2 will be developed. Each of the non-terminals is stored in a well-formed substring table or chart. This chart associates the non-terminals with their word