CHAPTER 4

ANALYSIS OF SYNTACTIC DETAILS

This chapter discusses the definite clause grammar and the modifications effected to improve the speed and modularity of parsing. The lexicon search procedure and the morphological analysis have also been described.

4.1 SYNTAX AND PARSE TREES

Syntax of natural language may be defined as the description of how words of the sentence group into patterns or phrases. There is a definite ordering on how the words combine and this ordering is called the grammar of the language. Grammar specifies a finite set of rules that decide how an infinite number of legal sentences may be formulated. Parsing is the method of applying this grammar to a sequence of words to produce a structured representation depicting the parse tree. The parse tree conveys the following information:

* syntactic word orderings
* individual syntactic constituents
* relationship between different syntactic constituents.

The syntactic parse tree and the corresponding list structure representation for the sentence 'The man eats' is shown in Figure 4.1.
FIGURE 4.1a SYNTACTIC PARSE TREE FOR THE SENTENCE "THE MAN EATS"

sentence(noun-phrase(determiner(the), noun(man)), verb-phrase(verb(eats))).

FIGURE 4.1b CORRESPONDING LIST STRUCTURE
4.2 DEFINITE CLAUSE GRAMMAR, ITS FEATURES AND ADVANTAGES

Rules of the definite clause grammar formalism are expressed as logical formulas of a particular kind called Horn clauses. Then the problem of parsing reduces to one of theorem proving. Horn clauses are a restricted type of first-order predicate logic and consists of two type of clauses:

* conditionless clause called a fact and
* conditional clause called a rule.

The notion that a collection of Horn clauses can be considered as a program (Kowalski 1979) was practically realized in the form of the programming language Prolog.

The data objects of the formalism are called terms. Terms can be a constant, a variable or a compound term. The basic unit of the formalism is the goal or the procedure call. The declarative semantics of definite clauses can be defined recursively as follows:

A goal is true if it is the head of some instance of a clause and that goal is true if the clause is a fact and each subgoal in the body of the clause is true for that particular instance of the goal. The instance of a clause is obtained by instantiating each variable with a new term for all occurrences of the variable.

Definite clause grammars while capturing all the advantages claimed by context free grammars such as clarity and modularity of expression, possibility of representing recursive embedding of phrases also provide three additional features. Definite clause grammar allows context dependency to be incorporated into the grammar by
adding an additional constraint defining subgoal, enables the syntactic structure to be constructed as the parsing proceeds and permits extra conditions to be checked whenever necessary (Pereira 1980, Gazdar 1989). The modified version utilized in this work uses a different modular mechanism for checking context dependent features such as number and gender agreement.

The procedural semantics of the definite clause grammar formalism used [in this case Prolog] determines the parsing strategy. Prolog executes grammar rules top-down, one at a time. Subgoals in a rule are executed strictly left to right. If a subgoal fails then backtracking is automatically performed, that is it rejects the most recently activated clause uninstantiating any variables that were instantiated by the activated subgoal. It then looks for alternative solutions to the subgoal that activated the rejected subgoal.

The general advantages of using logic programs for specifying grammar are:

* unification procedures are available for building representational structures and for passing information between procedures
* input and output are indistinguishable
* procedures can generate a set of alternative results
* procedures can pass as arguments incomplete structures which may contain variables that will get their values by calling other procedures (Warren 1978).

The advantages of definite clause grammar mechanism for interpreting and building syntactic representations are listed below:
* Perspicuity: The grammar of the natural language can be almost directly mapped to definite clause grammar since its declarative semantics allows definition of a language. The system is modular and is composed of small components or clauses which communicate only through explicitly passed arguments. Arguments in the clauses do not otherwise have any global meaning. Values of arguments once fixed cannot change except by uninstantiation controlled by backtracking.

* Power and Generality: Definite clause grammars provide an efficient, automatic and powerful mechanism for building structures. In addition the formalism is general enough to accommodate both the recognition and the generation process. Generality of the formalism allows the input and output formats to be of any form, list of symbols, strings, arbitrary tree structures etc.

* Conciseness: Definite clause grammar formalism are more concise because of the declarative format and the use of pattern matching unification process instead of the specification of explicit operations for setting and testing registers and building structures.

* Efficiency: Definite clause grammars are more efficient than other conventional formalisms due to the fact that it can be expressed directly in a general purpose programming language, Prolog. This eliminates the necessity of providing a separate compiler or interpreter.

* Flexibility: The definite clause formalism is essentially a language description and is neutral towards the particular parsing or execution mechanism and hence can be used for experimenting with different parsing strategies.
Suitability for Linguistic Work: Since definite clause grammar clearly specifies the definitions of the language it can directly be adopted by theoretical linguists who are concerned with the 'what' of language rather than the 'how' which is the main concern of computational linguists (Pereira 1980).

4.3 MODIFICATIONS TO THE ORIGINAL FORMALISM

The modified version of the definite clause grammar used in the system described in this work handles many requirements of the syntactic analyzer of natural language in a different manner. In addition to handling the construction of the syntactic structure in the same way as the definite clause grammar (Pereira 1980) the modified version builds a lexically detailed list containing all details obtained from the lexicon search procedure. This lexically detailed list describing the information obtained from the lexicon and any feature changing information derived from the morphological phase can be used by the ensuing stages of processing. The rules associated with the terminals of the natural language grammar have subgoals which activate the lexicon search procedure. The verification of the context dependent features such as number, gender and transitivity are performed employing separate subgoals [ instead of arguments as in the original version] utilizing the information obtained from the lexically detailed list available at each stage of processing. This system utilizes a special look-ahead heuristic which bundles and eliminates execution of unproductive rules, to improve the efficiency of search.
4.4 DEFINITE CLAUSE GRAMMAR AND THE SYNTAX OF ENGLISH

A typical context free grammar to represent a small fraction of simple sentences of English is given in Figure 4.2. We use definite clause grammar formalism to represent such grammars in a straightforward way by associating with each non-terminal, arguments to represent the start and end points in a string of a phrase represented as SO-S. Thus the above rules may be translated as shown in Figure 4.3. Taking rule1 as an example the logical formula may be explained as follows: the non-terminal sentence represented by the string SO-S consists of two non-terminals, a noun-phrase extending from SO to S1 and a verb-phrase extending from S1 to S (Clocksin 1981). The formula explicitly states that all the symbols in the string SO-S must be accounted for and a point S1 has to be discovered where S1 is the point of division of the sentence into two legal syntactic phrases. In order to satisfy the subgoal noun-phrase, another rule is called now with SO-S1 as the new sub-phrase. The parsing is essentially top-down where the sentence is successively broken down into smaller and smaller components until the terminal symbols are reached. The parser is initiated by calling the rule containing the definition of sentence with SO being instantiated to the complete sentence and S being made equal to the null list since the result of parsing should leave S as a null list by accounting for all the words of the sentence. If more than one rule exists to describe the same non-terminal then the rules are tried strictly in a top-down fashion that is the logical Or-ing is implicitly stated by defining alternative definitions. Here, the ordering of the rules play a major role. It is a good heuristic to order the rules such that the ones
Sentence \( \rightarrow \) Noun-phrase, Verb-phrase.
Noun-phrase \( \rightarrow \) determiner, noun/proper-noun.
Verb-phrase \( \rightarrow \) verb/verb, Noun-phrase.

**FIGURE 4.2 SIMPLE CONTEXT-FREE GRAMMAR**

\[
\text{sentence}(S_0, S) \rightarrow \text{noun-phrase}(S_0, S_1), \text{verb-phrase}(S_1, S).
\]
\[
\text{noun-phrase}(S_0, S) \rightarrow \text{proper-noun}(S_0, S).
\]
\[
\text{noun-phrase}(S_0, S) \rightarrow \text{determiner}(S_0, S_1), \text{noun}(S_1, S).
\]
\[
\text{verb-phrase}(S_0, S) \rightarrow \text{verb}(S_0, S_1), \text{noun-phrase}(S_1, S).
\]

**FIGURE 4.3 DEFINITE CLAUSE GRAMMAR FOR THE CONTEXT-FREE GRAMMAR**

\[
\text{sentence}(S_0, S) \rightarrow \text{noun-phrase}(S_0, S_1), \text{verb-phrase}(S_1, S).
\]
\[
\text{noun-phrase}(S_0, S) \rightarrow \text{proper-noun}(S_0, S).
\]
\[
\text{noun-phrase}(S_0, S) \rightarrow \text{determiner}(S_0, S_1), \text{noun}(S_1, S).
\]
\[
\text{verb-phrase}(S_0, S) \rightarrow \text{verb}(S_0, S_1), \text{noun-phrase}(S_1, S).
\]
with the smaller number of subgoals to be satisfied are placed first. While this may be a good general heuristic it was found that the handling of rules containing recursive calls to noun-phrase, verb-phrase and prepositional-phrase where the definitions are very much interleaved required the use of certain look-ahead information to improve the efficiency.

The terminals of the grammar are treated in a different way by this system. Here the procedure for searching the lexicon is initiated by the appropriate predicate contained in the rule for the terminal symbol. For example the subgoal determiner \([S_0,S_1]\) shown in Fig.4.3 calls the lexicon with the word to be searched extracted from the head of the list. After the lexicon returns the syntactic, semantic and feature details these details are checked to see if the word belongs to the required syntactic category and all the details are then passed as a list to the rule that contained this terminal.

4.5 BUILDING STRUCTURES AND FORMING THE LEXICALLY DETAILED LIST

Building a complete syntactic structure representing the sentence is an important aspect of the syntactic analyzer phase. As the parsing proceeds the structures are progressively built in the course of the unification process by using an extra argument to pass the structure from one level to the next. This is possible because the formalism allows structures that contain uninstantiated variables to be passed as arguments (Geetha 1990). In this work an extra argument is included to account for the progressive building of a
lexically detailed list that is instantiated for each of the terminal symbols and then appended together to form the complete list of lists containing all syntactic, semantic and morphological information extracted from the lexicon searching phase.

4.6 CONTEXT DEPENDENT FEATURE CHECKING

Context dependency features such as number and gender are checked using separate predicates instead of using the parameter passing on the spot checking carried out by the original version (Figure 4.4). In the formalism used here the feature information such as gender, number are readily available in the lexically obtained list structure and can be extracted and verified after the complete parsing procedure is over. This is advantageous in that it avoids unnecessary feature checking between constituents that are not going to form the syntactic constituents of the sentence.

Instead of having separate rules for transitivity checking the system employs a special restraining predicate to check the transitivity nature of the verb under consideration which is available in the lexically detailed list obtained during the course of parsing. The different moods of the sentence are tackled by first checking whether the sentence can be parsed as interrogative, imperative, covetive, etc.. If such rules fail, the sentence is considered to be declarative and parsing proceeds. This type of treatment is possible because of the modular and extensibility features of the formalism used.
FIGURE 4.4 SYNTACTIC PARSER AND IT'S INTER-ACTIONS
Passive sentences are tackled by using a look-ahead heuristic that checks to see if the given input sentence is indeed passive. The heuristic first checks whether the sentence can be considered a candidate for passive form parsing. The heuristic is based on the linguistic phenomenon that constrains the nature of the verb sequence that can be present in the passive form of the sentence. The presence of the 'be' form of the verb followed by a past participle form of a main verb indicates passive form. Thus the terminal symbols present in the sentence are checked beforehand to see if the above constraint is satisfied and the parsing is performed.

4.7 LOOK-AHEAD HEURISTIC

The system uses a word syntactic category based heuristic to avoid the unnecessary instantiation of unproductive predicates. The large number of alternative definitions required to define certain non-terminals of the grammar of natural language such as noun-phrase, verb-phrase, prepositional phrase and the recursive definitions of the adverb, adjective and prepositional phrases and the interleaved nature of some of the rules makes the search space to be considered quite large. The use of heuristics to prune this search space would go a long way in making natural language parsers efficient. In order to improve the efficiency of search, the system uses a heuristic look-ahead technique to perform this pruning. The concept used to design this heuristic is simple. The starting word categories of the different constituents constrain the different syntactic groupings to which these constituents can belong. By looking ahead at at most two or three word categories it may be
possible to eliminate the searching of the rules associated with that particular constituent. This is achieved in the formalism by adding a subgoal that conveys these constraints to be satisfied before the concerned predicate defining the syntactic non-terminal can be processed. This simple technique helps to reduce the search space by eliminating the rules associated with a particular syntactic phrase since the possibility of it not belonging to that syntactic type is checked in advance before actually executing those rules. The use of this type of limited look-ahead has been shown to improve the efficiency of parsing (Sampson 1983b).

4.8 MODULES OF THE SYNTACTIC ANALYZER

The input to the syntactic analyzer is a preprocessed list structure of words of the sentence. This list of words is obtained by using a string to list conversion predicate. This list which is the sentence list [SO] to be parsed is then passed on as a parameter to the main syntax analyzer predicate. The rules of the grammar with the necessary structure building and feature checking facilities are defined using the modified version of the definite clause grammar formalism described in the preceding section. The syntactic analyzer is initialized using the main predicate which uses the list structure [SO] as one of the input parameters. The output from this phase is the syntactic list structure and the lexically detailed list and the feature structure of the sentence. During the course of parsing the lexicon search modules and the morphological components are activated whenever necessary. The syntactic analyzer is executed in a top-down manner until the terminal symbols are
encountered. At this stage, a word of the sentence and its preferred syntactic category are available. Now the rule associated with the terminal symbol initiates the lexicon search procedure.

4.9 THE SEARCH OF THE LEXICON

The lexicon contains root words with associated basic syntactic, semantic and additional lexical based linguistic information. The syntactic category of the root word and all allowable syntactic categories that can be derived from the word are listed (Kuno 1986). The basic semantic details of the word are stored in the form of a set of semantic markers. A detailed description of the semantic information stored in the lexicon will be discussed in Chapter 5. The lexicon also associates implicit thread-bare structures conveying the illocutionary force in case the root word is a performative verb. The cohesional and temporal aspects if relevant are also associated with the root word. Interpretation of speech acts and cohesional and temporal aspects are discussed in subsequent chapters. The various components of the lexicon are shown in Figure 4.5.

The lexicon is organized in the form of a b-tree in order to improve the speed of search. A b-tree is simply a way to index the data in the lexicon. A b-tree is balanced in that the height of each branch of the tree is designed to be the same. The b-tree is made up of nodes and leaves. A node can have more than one branch and contains information for determining which branch the search should follow. This guiding of search information is available at all levels except the bottom-most. The leaves of the tree which are at the bottom level are the
Figure 4.5 The Components of Knowledge Contained in the Lexicon
actual terms or pointers to the terms. The leaf can contain more than one element in which case a simple comparison of each of the elements with the goal is performed. The utilization of the b-tree to organize the words of the lexicon in the lexicographic order improves the efficiency of search since fewer comparisons are needed to find a match (Figure 4.6).

The parser initiates the lexicon search procedure with the word whose details are to be found. If the required word is a root word and is available in the lexicon then all the details are collected and passed back to the parser in the form of a list. In case the word is not found in the lexicon, the morphological component is activated (Figure 4.7).

4.10 MORPHOLOGY AND THE MORPHOLOGICAL COMPONENT

The task of the module is to extract out the root word from the derivative of the root and in addition to pass on any feature information that may be conveyed by the derivative back to the lexicon search procedure.

4.10.1 Morphology

Morphological structure may be described as the internal structure of the words. Some of the constituency relationships that describe the morphological structure are affixation and modified forms. Many words are made of the affix and the root. The affix may be at the beginning [prefix] or at the end [suffix]. Typical affixes are 'un-', 're-', '-ed', '-ing', etc. The internal structure of words effect the syntax and meaning. Certain internal structure of words convey feature markings (Winograd
FIGURE 4.6 B-TREE
FIGURE 4.7 CALL OF THE MORPHOLOGICAL COMPONENT BY THE SEARCH LEXICON PROCEDURE
Thus the tense system of the verb [give, gave, giving, given], the number system of verbs [pay, pays], the number system of nouns [pencil, pencils], and the comparison system of adjectives and adverbs [soon, sooner, soonest] can be conveyed by the morphological structure of the words. Sometimes affixes can add to the meaning of a word [un-happy]. This function of the morphological structure is called compounding meaning. Affixes can even contribute to changing the syntactic category [create [verb] to creation [noun], child [noun] to childish [adjective], etc.]. Many morphological phenomena however exhibit a high degree of irregularity and hence cannot be described by a minimal set of rules. In addition to this irregularity in the form of the morphological structure, another irregular feature is the modification of the form of the affix depending on the word it is affixed to. For example consider the suffix -s,

-s Cy -> Cie (fries, tries but pays, toys)
Co -> Coe (potatoes, goes but zoos).

While it is possible to formulate a set of rules for many of the general cases of the morphological structure, exceptions have to be first taken care of (Parkison 1977).

4.10.2 Morphological Analysis

The morphological analysis can be multilevel in the sense that more than one type of affixation may be associated with a word, for example friend-li-ness and pay-ment-s. The module first analyzes the number feature associated with a word and then calls itself recursively for analysis until the root word has been found in the
lexicon or further morphological breaking is not possible (Pitrat 1988). All information about feature assignment, word-class conversion and semantic enhancement are passed back to the lexicon search phase. The weak inflectional system of English prevents the definition of the morphological rules in a systematic way. Information available about the preferred syntactic category from the parser and the allowable syntactic categories from the lexicon help the morphological component in choosing the correct rule for analysis. A typical morphological rule dealing with adverbs is outlined in Figure 4.8.

The root word obtained from the morphological component is again searched for in the lexicon. If the search succeeds then the details of the word are given to the parser otherwise it indicates the word has not been included in the lexicon or it is not a legal word (Figure 4.9).

The syntactic details of the word obtained from the lexicon search procedure is verified to find out whether it is of the required syntactic category and then a lexically detailed list is formed consisting of the word, its syntactic details, its root (if it is a derivative), its semantic details and feature information if any.
FIGURE 4.8 FLOWCHART FOR A MORPHOLOGICAL RULE
DEALING WITH ADVERBS
FIGURE 4.9 FLOWSHART FOR SEARCHING THE LEXICON
4.11 THE REPRESENTATIONS OBTAINED FROM THE SYNTACTIC ANALYZER

The outputs that are obtained from the syntactic analyzer phase are:

- a detailed syntactic list structure
- lexically detailed list of individual words of the sentence (Figure 4.10)
- feature structure of the sentence (Figure 4.11).

In addition to the above representations, it is possible by calling an appropriate predicate to obtain lexical details of independent root words and derived words.

An additional predicate has been defined that can obtain the words forming a particular syntactic constituent, the detailed syntactic sub-structure of a given syntactic constituent or the lexically detailed sub-list of the required component by specifying the path of the parse tree (Figure 4.12).

4.12 SYNTACTIC AMBIGUITY AND THE PARSER

During the course of parsing it is possible that syntactically ambiguous sentences are encountered. Syntactic ambiguity arises because the rules of the grammar does not provide a unique syntactic structure to the sentence but instead allows more than one structure to be associated with it. Since the parsing strategy adopted is top-down the initially occurring rules dictate the actual parse tree formed. The parser is so designed
FIGURE 4.18 OUTPUT OF THE SYNTACTIC PHASE FOR 'THE MAN EATS THE CAKE IN THE DISH'

SENTENCE: 'HAS THE HORSE BEEN SENT TO THE ZOO?'
MOOD: MAJOR INDICATIVE INTERROGATIVE
VOICE: PASSIVE EXPPLICIT AGENT
TRANSITIVITY: ACTION
THEME: POLARITY THEME
INFORMATION: UNMARKED

FIGURE 4.11 FEATURE STRUCTURE OF THE SENTENCE
'HAS THE HORSE BEEN SENT TO THE ZOO?'
PART OF THE SENTENCE
GIVEN THE PATH: (S,NNP,MP) --> (THE MAN)
(S,VUP) --> (LIKES THE GIRL)

SYNTACTIC SUBCONSTITUENT

FIGURE 4.12 SYNTACTIC PATH AND CORRESPONDING CONSTITUENTS
that the initially formed structure is taken as the only structure formed even if the sentence is syntactically ambiguous. This derived parse tree is maintained as such and passed on to the semantic processing stage. It is the job of the semantic analyzer to check for the validity of the syntactic structure and if found not compatible, the originally formed syntactic structure is restructured as per the dictates of the semantic constraints. This single predetermined structure formation eliminates the extra burden of backtracking and searching for alternative structures for even syntactically unambiguous sentences and allows ambiguity to be resolved at the time the information needed for disambiguation is available.