Universal Clause Structure Grammar

Thesis submitted for the award of the degree of
Doctor of Philosophy

K. Narayana Murthy

Department of Computer and Information Sciences
University of Hyderabad
December 1995

CERTIFICATE

This is to certify that the thesis work entitled **"Universal Clause Structure Grammar"** being submitted by **K. Narayana Murthy** in partial fulfillment of the requirement for the award of degree of Doctor of Philosophy (Computer Science) of the University of Hyderabad, is a record of bonafied work carried out by him under my supervision.

The matter embodied in this thesis has not been submitted for the award of any other research degree.

Dr. A. Sivasankara Reddy
Supervisor
Department of Computer and Information Sciences
University of Hyderabad

Dr. A. K. Pujari
Head
Department of Computer and Information Sciences
University of Hyderabad

Prof. V. Kannan
Dean
School of Mathematics and Computer/Information Sciences
University of Hyderabad

ACKNOWLEDGEMENTS

My thesis supervisor, Dr. A. Sivasankara Reddy, has been my teacher, my friend, my benefactor and my guide. As a teacher, he inspired as much as he instructed. He was always there to support me through rough times, and that is when it counts. Discussions with him used to leave me thinking for long, helping me to formulate my ideas clearly. It has indeed been a pleasure working with him.

I have gained a lot through the discussions I have had with Prof. Rajeev Sangal and Dr. Vineet Chaitanya of I.I.T (Kanpur) and Prof. Probal Dasgupta and Dr. G. Uma Maheshwara Rao of the Center for Applied Linguistics and Translation Studies, University of Hyderabad. Their views, comments, criticisms and suggestions have been very valuable to me. My special thanks to Uma Maheshwara Rao who read through the final draft in detail and gave me valuable, contentful, editorial and stylistic suggestions.

Vasudeva Varma deserves special mention. His comments on the first draft of my thesis helped me a lot in giving it a better shape. He has been using my parser for his research work and his periodic feedback has been very valuable. I must also thank Mr. Amit Sood and all my M.Tech. students who have tested my parser on a large number of sentences of their own. My thanks are also due, to all the teachers who participated in the fifth DOE sponsored NLP teachers' training programme here during December 1994. They were the first to see UCSG parser at work and offer a number of helpful suggestions.

I wish to thank Prof. A. K. Pujari, Head, Department of CIS and Prof. V. Kannan, Dean, School of MCIS, for their cooperation and support throughout. I have learnt a lot from all the members of the faculty in the department of CIS. They have been my teachers as well as my colleagues. Special mention must also be made about all the staff of A. I. Lab. for their timely help and cooperation. To all of you, my thanks for making it such a wonderful place to be in.

My parents sacrificed everything to give us education. Their inspiration has always been with me although they are physically far away.

Their was nothing like playing with Arabhi after long hours of intense work. My special thanks go to Nalini for her infinite patience and cooperation. Finally, I must thank Magnine for helping me to climb the infamous thesis hill by keeping my stomach free of activity.

K.Narayana Murthy

Abstract

This research work is about Natural Language Processing (NLP). In particular, it is about a theory of syntactic analysis of natural language sentences. We take a linguistic approach and develop a new grammar formalism called Universal Clause Structure Grammar (UCSG). Our primary aim is to develop a computationally viable grammar formalism that leads to simple grammars and efficient parsing for both positional languages and relatively free word order languages. We do not make any cognitive claims.

In UCSG we divide and conquer. By appropriately dividing the problem into subproblems we get to see where exactly relatively free word order languages differ from positional languages and what is really common between the two. We apply the least complex grammars that are sufficient for each of the subproblems thereby leading to very efficient parsing. Also, grammars for the subproblems can be written separately so that grammars become easy to write.

Grammars can be viewed as sets of constraints on the structure of sentences. There are constraints on linear position of constituents, hierarchical nesting of constituents one inside the other and functional dependencies between different constituents in a sentence. We find that these three primary kinds of structure inherent in human languages, namely linear, hierarchical and functional structures, lend themselves naturally for analysis by three independent modules. The three modules in UCSG are called linear (L), hierarchical (H) and functional (F) components. Both the grammar and the parser are divided into L, H and F parts. Correspondingly, there will be three levels of syntactic representation called L-Structure, H-Structure and F-Structure. While other grammar formalisms also tackle linear, hierarchical and functional aspects some way or the other, division of labour into these three separate modules, especially the introduction of an independent H level is the highlight of the UCSG formalism.

F-Structure depicts the assignment of functional roles to the various participants in a predication. The F part of the grammar provides constraints for role assignment. Functional roles should be so selected that role assignments can be made based mainly on syntactic information. We propose a set of eight functional roles motivated by the question-answer point of view. We employ a combination of top-down and bottom-up strategies for role assignment. The subcategorization frames associated with verbs provide the top-down expectations and the surface case markers associated with the nouns provide the bottom-up information. Any set of role assignments that satisfy all the F constraints simultaneously gives a valid F-Structure.

Functional role assignment is essentially local to a clause. Hence knowing the clause structure helps to localize the functional structure analysis to the respective clauses. The H part of the grammar specifies constraints on the hierarchical structure of clauses within a sentence. We show that the hierarchical structure of clauses in a sentence can be obtained before and without applying the functional structure constraints. We observe that there are strong constraints on the sequences of verbs and certain clause boundary markers called sentinels. We exploit these constraints to identify clauses, to determine the hierarchical nesting of clauses and even to determine the clause boundaries although only partially. H-Structure depicts this hierarchical structure of clauses. H grammars for different languages show only parametric variations even across positional and relatively free word order languages. The grammar of clause structure is highly language invariant. Hence the name Universal Clause Structure Grammar.

We observe that the atomic units of structure at both hierarchical structure and functional structure level are groups of words rather than individual words. We therefore propose that all potential word groups be identified first and both hierarchical structure analysis and functional structure analysis be carried out on sequences of word groups rather than on sequences of words. This reduces the effective length of the sentence and increases the efficiency of parsing. The L part of the grammar provides constraints for group formation. We show that the L grammar constraints in both positional and relatively free word order languages are primarily constraints of linear position. Hence the name linear structure.

We take care of inter-clause dependencies by working from whole to part instead of working from left to right. We determine the clause structure first and then analyze the functional structure clause by clause. We start from the main clause and go recursively into the nested clauses. We pass on information and expectations about displaced, shared and missing participants down the hierarchy of clauses. This makes functional structure analysis of a clause completely independent of the functional structure of any other clause in the sentence. There will no longer be any long distance dependencies.

We show that all potential word groups in a sentence can be obtained using finite state machine power in linear time. We also show that hierarchical structure analysis can be done using a very small number of context free grammar rules in cubic time, cubic in the number of verb groups and sentinels, which is typically much smaller than the number of words in the sentence. We further argue that hierarchical structure analysis can also be done in linear time if we bring to bear psycholinguistic and statistical evidence on the limitations of hierarchical structure of natural language sentences. Finally, we show how the localization of functional structure analysis to the local domains of the respective clauses makes parsing in UCSG highly efficient on the whole.

We make comparisons with other grammar formalisms and provide details of implemented parsers for English and Telugu. UCSG parsers produce only a small number of parses for a given sentence. The structural descriptions produced by the parsers clearly depict linear, hierarchical and functional structure inherent in a given sentence.

Contents

Chapter 1

Introduction

In this chapter we start our discussion with some background material that will be useful in specifying the boundaries of our problem clearly, in explicating certain assumptions often left tacit, and in clarifying some differences of views commonly held within related disciplines. We will first specify the aims and scope of this research work. After that we will attempt to justify the need for work of this nature by briefly sketching what is currently available and what is really needed. Then we will go on to take a glance at the proposed theory and we will see sample transcripts from the implemented system. We will summarize the main claims and results obtained. We will end this chapter with an outline of how the rest of the thesis is organized.

1.1 Introduction

The research work presented in this thesis is about Natural Language Processing (NLP), also known as Computational Linguistics (CL). The subject matter in NLP is natural languages - languages which human beings use to communicate with each other. Communication does not mean communication of information alone. There is a much broader significance to the term communication. People use language to communicate information, attitudes, feelings and intentions and to bring about a variety of effects in the minds of the listeners.

1.1.1 Natural Language Processing

NLP takes the view that language is used mainly for communication, in this broad sense, between active cognitive processors [52]. NLP has endeavoured to build computational models of the mental processes of the listener/reader and the speaker/writer. What made the speaker say this, why did he say it this way, what does he expect the listener to do, what does the listener really understand, how does he relate this to the knowledge he already possesses and how does he react to this new piece of information, are all in fact questions relevant to, and within the scope of NLP. In NLP we need to model the goals in the minds of the speakers and listeners, their strategies and plans for achieving these goals, the knowledge they possess about the situation and about the world in general as well as the planning and reasoning processes that go on in their minds. The characterization of the commonsense knowledge of the world, ability to store, index and efficiently retrieve the stored knowledge when required, and the ability to apply the relevant knowledge for reasoning, are all thus relevant to NLP. For this NLP interfaces with its sister disciplines within Artificial Intelligence (AI) such as knowledge representation and automated reasoning.

In building computational models the orientation could be towards modelling how exactly humans understand, generate or learn a language. This orientation is typical of linguists, psychologists and cognitive scientists. Alternatively, the orientation could be more towards building computationally viable and efficient models, borrowing from available knowledge of human language processing but not being dogmatic about it. This second, more practical view is typical of engineers and Artificial Intelligence researchers. The two orientations are really not contradictory but complementary to one another. In this work we focus on developing computationally viable and efficient models without directly making any cognitive claims. However, we do share the common belief that the human brain employs simple, elegant and efficient methods. Thus, if our model turns out to be simpler and more efficient than other available models we think there will certainly be significant cognitive implications.

There are different possible approaches to NLP including linguistic, statistical, connectionist and formal logic based approaches. We subscribe to the linguistic approach. In this approach we develop explicit grammars that specify the structure of language. We believe that statistical and connectionist approaches are better used as embellishments for enhanced flexibility and robustness than as substitutes to the more direct and explicit linguistic approach.

NLP has traditionally restricted itself to written text leaving out all aspects of speech processing to a distinct discipline. This division of labour is primarily motivated by the technological differences between text processing and speech processing. Recently, there has been an increasing tendency to club traditional NLP applications with speech. In this research, we have limited ourselves to written text processing only. By written text we really mean the text that is entered into a computer through the keyboard, not handwritten cursive script.

Natural Language Understanding

NLP has endeavoured to build computational models of the mental processes of the listener/reader and the speaker/writer, respectively termed Natural Language Understanding (NLU) and Natural Language Generation (NLG). The final aim of a Natural Language Understanding system is to get not only the meaning but also the possible intentions behind the utterance. Similarly the aim of a NLG system is to synthesize an utterance that will hopefully convey the intended information and also bring about the intended effects in the mind of the listener. Thus NLP takes a broad view and includes within its scope lexical, morphological, syntactic, semantic as well as pragmatic aspects of language.

The focus of this work is on Natural Language Understanding. In NLU written sentences are taken as input and a formal, symbolic representation of the meaning of the utterances is to be produced as output. The process of mapping from the inputs to the outputs is termed *the process of understanding*, or simply *understanding*. The process of understanding makes use of knowledge of the language, knowledge of the context in which the utterance was made as also general knowledge of the world. In this thesis we restrict ourselves to syntactic analysis of natural language sentences.

1.1.2 Syntax

Mapping a given utterance to its meaning representation is a very complex process. In a computational model, a detailed and precise algorithm is needed to specify how exactly this correspondence between an utterance and its meaning representation is established. Languages exhibit complex structures and a detailed and systematic analysis of the structure of natural language sentences is invaluable in determining the meaning of the sentences. Syntax deals with the problem of analyzing the structure of natural language sentences and producing structural descriptions. These structural descriptions can then be analyzed further by semantic and pragmatic components to obtain the meaning and the intentions behind the sentence thereby capturing the communicative aspect of language in its broadest sense. The focus of this research work is on syntactic analysis. We develop a theory of syntactic analysis of sentences in natural languages.

While it is well recognized that people do not communicate using isolated sentences but with coherent pieces of discourse, most syntactic theories limit themselves to handling one sentence at a time, ignoring interactions between sentences. Firstly, this is possible because sentences have their own internal syntactic structure that is for the most part independent of how and where the sentence is used. Secondly discourse level grammars have not matured to a stage where one can employ discourse grammars in practice for syntactic analysis. In this work we therefore presume that the input to the syntactic analyzer is one isolated sentence.

In a sense, the structural descriptions produced by the syntactic analyzer can themselves be viewed as very crude and raw representations of meaning, only waiting to be refined and perfected by semantic and pragmatic components. These structural descriptions only aim to capture the raw information content in the sentence. The structure of a sentence is analyzed in terms of meaningful parts and subparts and the relationships among these parts. The smallest parts are the words which can be mapped on to grammatical categories. While within a syntactic framework we normally deal with grammatical categories like noun and verb as uninterpreted symbols, it should be emphasized that these categories are also essentially semantic in nature - nouns represent *things*, verbs represent *actions* and *states*, etc. When we say that 'book', 'beauty' and 'London' are all nouns, we are emphasizing what is really common between all these entities - that they are all *things* in a broad sense. Thus grammatical categories are *general meanings*, very general of course. When we need to make somewhat finer distinctions, we use the notion of subcategorization. We divide nouns into proper nouns and common nouns, verbs into intransitive, transitive and bitransitive verbs and so on, giving slightly more detailed semantic classification. In any case, syntactic analysis is a kind of *understanding* wherein very gross, general meaning is captured. The syntactic structures that are obtained in such an analysis are representations of such very abstract and general meaning.

Syntax as an Independent Module

In NLU syntactic analysis is a step towards the final goal of understanding the meaning and it is not appropriate to deal with syntax independent of the final goals of NLP. What kind of analysis needs to be made and what kind of structural description should be produced are therefore dictated by the needs of the subsequent semantic and pragmatic processes. Syntax for its own sake is not justified in NLP. This is not the sense in which *autonomy of syntax* is to be taken. The question is whether syntax can be taken up and studied as an independent component within the overall system.

Some researchers have argued that human mind does not process sentences in separate levels such as syntax, semantics and pragmatics and hence a computational model must also employ an integrated approach to language understanding. Roger Schank and followers from the Yale University have been stressing integrated processing from a long time. See for example, [43, 6]. Others, mainly linguists, have argued and even attempted to provide evidence for the psychological reality of independent modules.

We would like to emphasize that integrated or modular processing approaches should not be confused with implementation strategies. The question is not really whether syntax, semantics and pragmatics should all be done together or one after the other - this is more of an implementation strategy. The real question is whether it is reasonable to break the task into separate and independent modules. The degree of interaction between various modules is the dictating factor. The proponents of the integrated processing approach believe that the different parts interact so much and in so many complicated ways that it is not reasonable to break the task into separate modules. Others believe that it is possible to break the task into nearly decomposable modules. By *nearly decomposable* we mean that the interactions between the modules are minimal, localized and clearly understood so that we can treat the modules essentially independent of one another and take care of the interactions separately.

Of course if relevant semantic and pragmatic information was available, it could help the syntactic analyzer. Is this worth the effort and is it practically possible in all cases? To illustrate the nature of this situation, we will use an analogy. Expanding contractions such as *can't* and *won't* is essentially a morphological level process. However to expand the contractions in the following sentences

1) He's a nice boy
2) He's done it!

one has to get into the level of syntax. More interestingly, we will have to get into the level of discourse analysis if we have to determine whether the *we're* in the following sentence is *we are* or *we were* since the tense can be determined in this case only based on discourse coherence.

3) We're eagerly waiting for the arrival of our new car.

Should we do discourse analysis before, or at the same time when we are trying to find out what the words in a sentence are? We feel it is much better to simply note down that *we're* can be either *we are* or *we were* and proceed with morphological and syntactic analyses. If the effort involved in solving a problem is too much, it is wiser to see if the problem can be left unsolved at this stage, hoping that at a later stage the necessary information would become more readily available. Postponing the solution of a problem can make it much more simpler, thereby more than compensating for the extra effort involved in keeping track of the unresolved problem all that way along. This is the sense in which we take *autonomy of syntax*. We will deal with syntax as independently as possible and we will simply postpone decisions that require extensive or costly processing at the semantic or pragmatic levels.

It is not always possible or desirable to try to obtain all the required information to resolve an ambiguity. Obtaining all required information, even when theoretically possible, can be extremely costly in itself thereby more than compensating for the gain in being able to make the current decision correctly. Consequently, we believe that total determinism as claimed by Marcus [31] is not a feasible goal. Yet it is highly desirable to try to attain determinism to the maximum extent possible. Determinism means being able to take the right decisions at the right time and is therefore both theoretically and practically highly desirable. This requires NLP to be viewed in terms of requirements, availability and flow of information and breaking the problem into modules accordingly. We view syntax as an independent module, but incorporate a small amount of semantic knowledge that can greatly reduce the number of different parses produced.

We take the modular approach and propose an independent syntactic analyzer. Our model demonstrates that it is possible to deal effectively with the structure of natural language utterances using only a bare minimum of semantic information. It should also be noted that our current understanding of semantics and pragmatics is no way as clear and complete as our knowledge about syntax and hence even if integrated processing may finally turn out to be the ideal approach, a modular approach may be the best that one can follow for the time being.

1.1.3 Grammars

We have defined syntactic analysis as a process of mapping a given sentence to a description of its structure. The inputs to the syntactic analysis process, also called the parsing process, include the given sentence and the *knowledge of language*. Knowledge of language includes, among other things, lexicon, morphology and grammar of the language. Grammar of a language is a formal specification of the valid structures in that language. Since our focus is on syntactic analysis, we use the term *grammar* in this limited syntactic sense. Since we have taken an NLU orientation, we develop our grammars from an analysis point of view rather than from the point of view of synthesis. It may be noted that linguistic theories normally discuss grammars from the point of view of *generating* sentences.

Process View and the Importance of Efficiency

It is a central working hypothesis of the linguistic models that a nonprocessing characterization is desirable. Grammars can then be developed independent of how they are going to be used by parsers and generators. Details of the algorithms can be kept separate from the basic characterization of the knowledge of language. Grammars can be viewed as a specification of a space of grammatical possibilities that does not say anything about how to search that space. While this division of knowledge into *what* and *how* is a boon to the grammar writer, who can write his rules without worrying about the details of the parser, a purely abstract specification oriented view of grammar can be a very bad choice in NLP.

While linguistics has typically dealt with characterization of structures, the science of computation deals with theories of processing. NLP attempts to build computational models of the processes of understanding and synthesis. Computational paradigm is based on the belief that by developing idealized theories of the processes involved, we will have a clear and revealing way of explaining the structure. The structures follow from the processes.

One should be careful not to confuse between idealization and proceduralization. One should not mix up the issue of procedural versus nonprocedural approach with the distinction between competence and performance. Chomsky argues for a model of competence and against a performance model by talking about memory limitations, changes in intention during speech, and even physical states such as coughing, sleepiness or drunkenness. See for example [10]. Nobody for sure would want to build a model of performance that includes coughing and sneezing. Idealizations are done in all approaches. This is not any significant argument against procedural view at all.

Ignoring all purely physical and psychological influences of actual performance, the remaining *knowledge of language*, the characterization of the mental competence of a speaker, also has a procedural part in it that has got to be dealt with explicitly. The knowledge of language includes not only rules, constraints and principles but also the procedural aspects of what rules, constraints or principles are applied, how, when, in what order, etc.

It should be emphasized that knowledge does not always have to be declarative and nonprocedural. The *procedure* for multiplying two numbers is very much a part of the *knowledge* of most of us. There was a time in the history of AI where the arguments for and against procedural and declarative representations of knowledge had taken the shape of a controversy. It is universally agreed now that both kinds of representations of knowledge have their due share. There is no clash between the two, one complements the other.

There is no guarantee that a good purely declarative competence model automatically leads to efficient performance. A model cannot be judged purely based on the structure of the declarative grammar that it employs. A model would be good only if both the declarative and procedural components are good. Some of the models developed by linguists, such as Government and Binding, when implemented in a straightforward way on a computer, will be extremely inefficient. If we all agree that human beings are efficient processors of language, how can we consider these models as good cognitive models? Grammars must be inherently amenable for efficient parsing. Grammars must be *designed* for efficient processing.

1.1.4 Grammar Formalisms

Every grammar defines a language but a given language can be generated by many different grammars. All possible grammars of a language will not in general be equally simple, natural and amenable to efficient parsing. Also, grammars can be written in a variety of ways. Phrase structure rules, trees and tree operations, principles and constraints are commonly used for writing grammars. Particular formats in which the specifications for valid structures in a language are given can affect the elegance and simplicity of the grammar, the ease of grammar writing for different human languages, and the suitability of the grammar for efficient processing.

We therefore need a meta language, called a *grammar formalism*, to specify how exactly the grammar of a language is written. A grammar formalism is a meta language - it restricts the kinds of grammars that can be written, it defines the structure of a grammar, and hence the properties of the syntactic system employing that grammar. A grammar formalism also specifies the nature of processing that is involved in syntactic analysis and the kinds of structural descriptions produced. All grammars written within a grammar formalism will have such properties imposed on them by that grammar formalism. Thus instead of talking about the desirable properties of individual grammars of specific human languages, we can talk about these properties at the level of the grammar formalism itself. Hence the design of a good grammar formalism can be taken as the central problem in syntax. In this thesis we have endeavoured to develop a new grammar formalism which we have called Universal Clause Structure Grammar (UCSG). Before we say more about our grammar formalism, let us see what constitutes a good grammar formalism.

What Constitutes a Good Grammar Formalism?

Computational models require exhaustive grammars for the languages being handled. Writing exhaustive grammars is not a simple task. A good grammar formalism requires only a small number of simple grammar rules thus simplifying the grammar writing task. A simpler grammar would also be a better choice as a cognitive model than a grammar that employs unnecessarily complex contrivances. Examples of such artifices can be seen in the special mechanisms employed in many grammar formalisms to deal with long distance dependencies.

A smaller and simpler grammar also makes the parser very efficient because at each step there are only a few alternatives to consider. An efficient parser is required not only because of practical needs for fast processing but also from theoretical point of view. An efficient syntactic system is a better candidate as a model of human language processing than an inefficient one. Since less powerful grammars will in general lead to more efficient parsing, the search in NLP is for the least powerful grammar that is sufficient to deal effectively with natural languages.

It has been well recognized that despite superficial differences, human languages share certain common underlying principles. Thus theory is on the look out for a universal grammar - a grammar that uncovers these underlying universal features. Our use of the term universal grammar is different from the *Universal Grammar (UG)* as Noam Chomsky and other linguists have used [11]. For them the UG is an in-born biological endowment of the human mind. We make no reference whatever to the human mind or what things are pre-wired into the human brain and what things are learnt after birth. We deal with language as people use it, not with the hypothetical language or grammar that may be there inside the mind of humans. Our use of the term 'universal' only implies the identification and separation of language independent aspects of grammar from the idiosyncratic features of individual human languages. A grammar that is largely common across human languages is theoretically very desirable even without any reference to the human mind [22]. Also, in practice, recognition of these universal and innate properties of human languages helps in building grammars and parsers for various human languages with a minimum of effort, with a minimum of duplication of work.

A good grammar formalism should focus on the universal aspects of grammar rather than on the idiosyncratic features of specific languages. The idea is to concentrate on the rule rather than on the exceptions. It is not as important to deal with the idiosyncratic, peculiar and infrequent constructions in specific languages as it is to deal with the more usual, common and frequent constructions elegantly and efficiently. Knowing the rule from the exception is extremely important for getting insights into the true nature of human languages, for discovering the universal nature of our languages.

Let us now turn to the structural descriptions produced through syntactic analysis. One of the most commonly used representations of syntactic structure is the tree. The children of a node in a tree are also trees by definition. Thus trees show the nested, part-whole, or hierarchical relationships between different constituents. Trees used in NLP and linguistics are ordered trees - the linear order of the nodes in the tree is significant. Trees thus show linear as well as hierarchical structure of sentences.

In many languages, the linear order in some nodes is significant while the linear order among the other nodes is insignificant. These languages are called relatively free word order languages. Indian languages belong to this type. In a Telugu sentence, for example, all possible permutations of nouns are syntactically valid but typically the verb can only come at the end. Also while the nouns in a clause can come in any order, all of them must necessarily come linearly before the nouns in the following clause. Linear order is important in some cases and insignificant in others. Trees have difficulty in coping up with this variation.

Apart from depicting the linear and hierarchical structure of sentences, a syntactic analyzer, we believe, must also determine the roles played by the various constituents in the sentence. The functional structure of a sentence depicts the assignment of functional roles to the various constituents in the sentence. Role assignment depends on functional structure constraints such as linear position, morphological inflections, agreement, subcategorization and selectional restrictions. A good structural description must clearly show the functional structure of the input sentence. Trees cannot depict any kind of functional dependency. Trees are often augmented with special links connecting various nodes in order to indicate functional dependencies. The resulting structures are really no longer trees but much more complex structures called graphs.

Trees have several other demerits. A tree is a monolithic structure that includes units at different levels. A tree is a mess of words, phrases, clauses and the entire sentence. Consequently, parts which logically form one group are thrown far apart. All the problems of long distance dependencies originate from this. Also, trees tend to become very large and unwieldy for long and complex sentences. Every elementary subtree, that is, a subtree that includes only one node and its children, corresponds to one application of a phrase structure rule. Thus trees are closer to phrase structure rules than to the structure of the sentences. We need structural descriptions which separate out and vividly show linear, hierarchical as well as functional structure inherent in the given sentence. Trees are not suitable for depicting the structure of natural language sentences.

In summary we consider a grammar formalism to be good if it permits simple, elegant, computationally viable and universal grammars to be developed with ease and efficient parsers to be developed within the formalism. Also, the syntactic descriptions produced should clearly depict linear, hierarchical and functional structure inherent in the given sentence.

We will now briefly sketch the merits and demerits of existing grammar formalisms. Keeping the desirable qualities of good grammar formalisms on the one hand and the strengths and weaknesses of available grammar formalisms on the other, we will then endeavour to develop our own grammar formalism - the UCSG formalism. Let us first understand the strengths and weaknesses of Context Free Grammars (CFG). This would help us in understanding other grammar formalisms clearly.

1.2 Context Free Grammars and Natural Languages

In the 1950s, there were major developments taking place in the field of computer science, including the development of high level programming languages. Grammars and parsers needed for compiling these programs received a great deal of importance. In his papers in 1956 and 1959, Noam Chomsky laid out the foundations of formal properties of grammars [8, 9]. Chomsky classified grammars based on string rewriting rules into four classes of formal complexity called Type 0 or Unrestricted Phrase Structure Grammar, Type 1 or Context Sensitive Grammar, Type 2 or Context Free Grammar and Type 3 or Regular Grammar. Context Free Grammars (CFG), the second least powerful in the Chomsky Hierarchy, were found to be ideal for dealing with programming languages efficiently. Since then a great deal of formal studies have been made on CFGs within computer science and very efficient parsing algorithms have been developed [54, 15, 49].

In the sixties, CFGs were also applied to the study of natural languages. As Woods [53] puts it in his 1970 paper, "Context-free grammars have been immensely successful (or at least popular) as models for natural language in spite of formal inadequacies of the model for handling some of the features that occur in existing natural languages". There have been many papers arguing for or against Context Freeness of natural languages [38, 46, 24, 32]. In a formal sense CFGs have sufficient weak generative capacity to deal with almost all aspects of natural languages, the only formal limitation being its inability to handle cross serial dependencies. In a practical sense, however, CFGs have several disadvantages.

The merits and demerits of CFGs for dealing with human languages are now fairly well understood and it is generally accepted that CFGs do not form a good model for natural languages. Yet the study of CFGs is significant in NLP for four reasons. Firstly, significant portions of natural languages *are* context free. Since our aim is to develop a grammar formalism of the least powerful type, it is important that we understand the strengths and weaknesses of CFGs. We should not employ more powerful grammars where CFGs suffice and we should clearly understand the reasons for using more powerful grammars where really required. Secondly, most of the other grammar formalisms also incorporate CFGs in some form or the other, at some stage or the other. Thirdly, some of the important syntactic phenomena like long distance dependencies, cross serial dependencies and parsing of relatively free word order languages are also directly related to the powers and limitations of phrase structure rules in general and CFGs in particular. Lastly, development of other grammar formalisms can be fruitfully viewed as realization of the weaknesses of CFGs and various ways of reacting to these limitations. Keeping these points in mind, we first take up a detailed study of CFGs for natural languages.

There is a direct relationship between the generative capacity - the kinds of sentence structures which a grammar can generate or parse, and computational complexity. The most general grammar takes the largest amount of computational resources (mainly processing time and/or memory space), and the least general grammar can be parsed with the least computing resources. It is important to note that these differences in computational complexity are not small differences. They are orders of magnitude differences. It has been natural in NLP, therefore, to look for the least powerful grammar that is sufficient for dealing with natural language sentences.

We might therefore ask ourselves whether regular grammars, the simplest type of grammars within the Chomsky Hierarchy, can be used to describe human languages. Regular grammars can handle all finite languages and also some infinite languages characterized by repeated substrings. Natural languages are commonly taken to be infinite and they involve more than mere repetition of subparts. Regular grammars are not sufficient to capture the whole of natural languages. A formal proof is based on the fact regular sets are closed under intersection [20].

The next question would therefore naturally be whether context free grammars can be used. There are deterministic subclasses of CFGs which can be parsed in deterministic linear time. Even parsers based on general CFGs have a worst case time complexity of $O(n^3)$ where 'n' is the number of words in a sentence. The question of context freeness of natural languages has received a lot of attention. See the book entitled 'The Formal Complexity of Natural Language' [51] for example. As the editors of the book have rightly noted in the beginning of part two of their book, almost all the arguments given to demonstrate the non-context freeness of natural languages have hinged around peripheral, rare and certainly not the run-of-the-mill language phenomena. Also, as Gazdar and Pullum have pointed out in their paper in that book, many of these arguments are neither technically nor empirically sound. They contend that nobody has yet given a real proof that natural languages are not context free.

Whether natural languages are fully context free or not, is not really very important. The point of importance is, all these arguments and counter-arguments notwithstanding, *natural languages are largely context free.* We may recall that computer programming languages also have some features that are certainly not context free and yet context free grammars have been applied very fruitfully to deal with these languages very efficiently. We need to clearly understand which aspects of the syntax of natural languages are really context free and which aspects are not. We will then be able to apply more powerful grammars only where they are really required. Parsing with CFGs will be very much faster than parsing with more complex grammars. In fact, there are parts of natural language syntax that do not even require context free power, regular grammars are sufficient. What is really needed is an appropriate way of breaking the problem into subproblems and employing the least powerful grammar that is essential for each. This has been the guiding principle in our development of the UCSG grammar formalism.

In a formal sense CFGs have sufficient weak generative capacity to deal with almost all aspects of natural languages. Natural languages are largely context free. In a practical sense, however, CFGs have several disadvantages. We will now look at specific syntactic phenomena to highlight these limitations of CFGs.

1.2.1 Functional Dependencies

Languages enforce certain dependencies between the constituents in a sentence. Examples of this are grammatical agreement requirements and selectional restrictions. Thus 4) is grammatical but 5) is not, 6) is grammatical while 7) is not.

4) The students write the test
5) *The student write the test
6) The student read the paper
7) *The paper read the student

If we have to enforce these dependencies using (only) CFGs, we will be faced with two problems. For one thing, we will be forced to introduce new categories since CFGs, like all other phrase structure rules, can only generate strings of terminal symbols. (Note that we normally take grammatical category symbols as the terminal symbols when we write grammar rules.) There would be singular-nouns and plural-nouns, singular-verbs and plural-verbs. There will be nouns that denote *things that can read* and nouns that denote *things which can be read.* This would cause a multiplication of rules leading to very large rule sets. There would be, for example, separate rules for singular sentences and plural sentences. Computationally the syntactic system becomes extremely inefficient. Secondly, and more importantly, a simple requirement like agreement between subject and verb, which can be given in one simple statement in English, is being turned into a multitude of otherwise unrelated categories and rules. Simple generalizations are lost.

A word should be associated with a particular category purely based on its own intrinsic properties. A grammatical relation between two independent words is an intrinsic property of neither of these words. The phrase structure rules have no direct way of specifying these relations. Thus CFGs, all types of phrase structure rules for that matter, are unsuitable for dealing with dependencies of any kind between different constituents in a sentence. Also, the structural descriptions which CFG rules naturally generate, namely trees, cannot depict functional dependencies directly. It should be emphasized that we are not asserting that CFGs are insufficient. CFGs do have the necessary generative capacity to generate valid and only valid strings as far as such dependencies are concerned. But that is simply not the right way of doing things. Functional dependencies must be separately and explicitly specified in the grammar and clearly depicted in the structural descriptions.

1.2.2 Long Distance Dependencies and Movement

We have seen that linear order of words and phrases in a sentence may be significant. Grammars must therefore impose constraints on linear positions as required in the given language. Western grammar formalisms have been strongly influenced by languages like English where linear order constraints are very significant. Linear order has become an all important aspect of dominant linguistic models. There are valid syntactic constructions, however, where there are deviations from the usual order. Therefore these formalisms posit a *normal* or unmarked order and view syntactic constructs which deviate from the normal order as involving *movement*. The grammar is expected to specify what constituents move from which place to which place and under what conditions. No wonder this aspect forms quite an important and complex part of these theories.

Some movements are local or bounded, as in the case of auxiliary shift in simple yes-no questions (example 8) and subject-object inversion in passives (example 9).

8) Can the company sell Pentiums now?
9) My Pentium was replaced by someone

Other types of movements take place across arbitrarily long distances and are termed *long distance dependencies*. Relative clauses (examples 10 and 11) and wh-questions (examples 12, 13, and 14) in English are examples. In these constructions a constituent from within a clause will have been removed and moved to the left. Long distance dependencies have been one of the more serious problems for syntactic systems and various grammar formalisms have developed special techniques for dealing with them.

10) The machine which the company sold __ to me was defective
11) The machine which the company, which I believe everyone trusts, sold __ to me was defective

12) Which computers does the company sell __?
13) Which computers does the company which I believe everyone trusts sell __?
14) Which computers do you think the company which I believe everyone trusts sells __?

Historically, the concepts of movement and other kinds of transformations were introduced in order to account for the fact that sentences superficially looking different often have roughly the same meaning. So a *deep* structure, something closer to meaning than the surface structure of a sentence, was taken as the starting point and other related structures were obtained through transformations. It was soon shown that many of the supposedly meaning preserving transformations did in fact risk meaning changes and linguists had to give up the idea of linking deep structure to meaning. Linguists forgot the meaning aspect but dogmatically stuck to the notions of deep and surface structures, now renamed D-Structure and S-Structure, and movement continued to be the primary mechanism of going from the deep to the surface structure.

The notions of movement and other transformations are direct consequences of the employment of phrase structure rules and the corresponding ordered trees for describing the syntax of natural languages. As we have already seen, a tree is a mess of different levels of meaning units - words, phrases, clauses and sentences. In terms of linear position, the parts of a single meaning unit can be widely spaced apart. Long distance dependencies are only one type of functional dependencies and CFGs cannot effectively deal with any kind of dependency between two constituents.

1.2.3 Relatively Free Word Order Languages

There are a number of human languages where there is considerable, though not unlimited scope for changing the order of words in a sentence without significantly altering its basic meaning (to be more precise, without altering the functional structure of the sentence). Indian languages are examples of this. All of the following Telugu sentences mean essentially the same thing

15)

raamuDu	baDiki	siitatoo	veLLaaDu
Rama	school-to	Sita-with	went

16)

baDiki	raamuDu	siitatoo	veLLaaDu
school-to	Rama	Sita-with	went

17)

siitatoo	raamuDu	baDiki	veLLaaDu

Sita-with Rama school-to went

The strong point of CFG is that it can effectively deal with both the linear and hierarchical structure inherent in human languages. This ability to deal with linear structure comes back as a weakness when we have to parse sentences in relatively free word order languages. Here a variety of word orderings are allowed and *linear order is not to be considered significant.* CFGs however, have no way of getting rid of their hold on linear order. A CFG rule such as

s $\rightarrow$ np1 np2 vp

asserts two independent things. Firstly it asserts that s has three constituents named np1, np2 and vp. Secondly, it also asserts, unfortunately, that np1, np2 and vp have to come in that linear order. Thus the above rule is completely different from

s $\rightarrow$ np2 np1 vp

Since both of these orders can be equally valid in some languages, we must have both the rules in the grammatical system. For rules with more number of symbols in the right hand side, many permutations may be possible and we need to include one rule for each possibility. We are in the unhappy state of being forced to take something irrelevant seriously. Having one rule for each possible permutation is like imposing a serious restriction and then having one case for each way of relaxing that restriction. We multiply the number of rules thereby making our parser very inefficient. Note again that CFGs *are sufficient* in the minimal technical sense of generative capacity but practically just unusable.

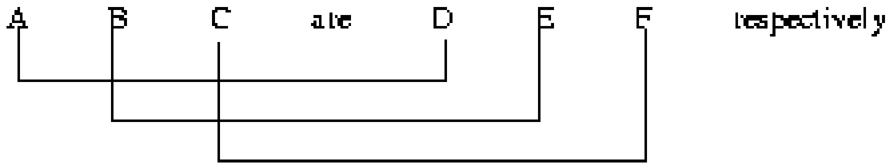
1.2.4 Cross Serial Dependencies

CFGs can handle constituents that come linearly one after the other as well as constituents that are properly nested one inside the other. By proper nesting we mean that an inner constituent must end before the outer constituent closes. In English, we find some examples of cross serial dependencies, as illustrated by 18)

18) Narasimha Rao, Shankar Dayal Sharma, and Chandrababu Naidu

are the Prime minister of India, the President of India and
the Chief Minister of Andhra Pradesh respectively.

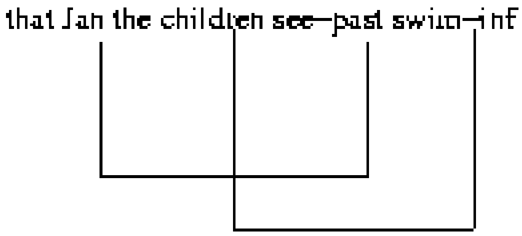
The 'respectively' construct 'A, B, C are D, E, F respectively' links A to D, B to E and C to F in a cross serial manner:



Cross serial dependencies occur in programming languages too. One example is in the requirement that the arguments must match one-to-one in a function definition and function call. Compilers simply ignore this aspect while doing the syntactic analysis of the source program and postpone the checking to a later phase. We could do the same thing for constructs like 'respectively' in English by simply taking 'A,B and C' and 'D, E and F' to be single composite constituents. It would be all right to leave out a few special constructions from syntactic analysis.

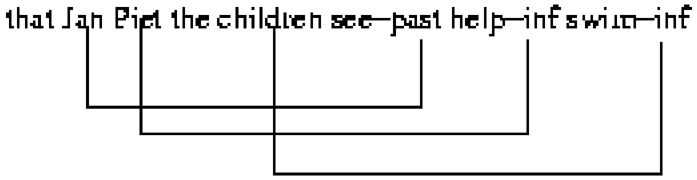
There are languages such as Dutch, however, where we come across an infinite set of grammatically correct sentences with cross serial nesting. A way of directly dealing with them would therefore be preferable. Let us look at some examples from Dutch. Both the examples and the discussion that follows are based on [7].

19) ... dat Jan de kinderen zag zwemmen



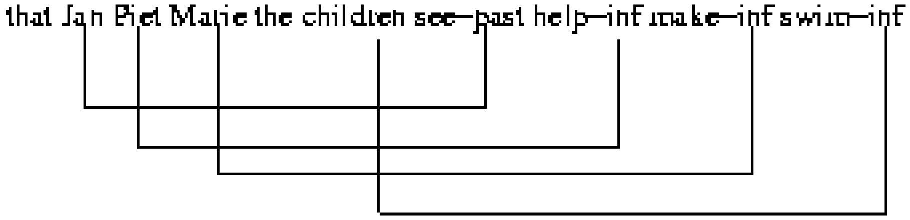
...that Jan saw the children swim

20) ... dat Jan Piet de kinderen zag helpen zwemmen



...that Jan saw Piet help the children swim

21) ... dat Jan Piet Marie de kinderen zag helpen laten zwemmen



...that Jan saw Piet help Marie make the children swim

There are restrictions on cross serial nestings in Dutch. The number of nps should match the number of verbs. The first np and the first verb must satisfy the basic agreement requirements. The final nps and the last verb must satisfy the subcategorization restrictions. Only a verb that is subcategorized for both a np and an infinitival complement without the complementizer *'te'* can be inserted in this cross serial fashion. There are only a finite number of such insertable verbs but they can be repeatedly used and hence there is no limit on the depth of such a nesting. It is to be noted that within these restrictions, all arbitrary permutations of the nps in the np sequence and all arbitrary permutations of the verbs within the verb sequence produce grammatical sentences:

22) ...dat Jan Marie Piet de kinderen zag helpen laten zwemmen
...that Jan saw Marie help Piet make the children swim

23) ...dat Jan Marie de kinderen Piet zag helpen laten zwemmen
...that Jan saw Marie help the children make Piet swim

24) ...dat Jan Marie de kinderen Piet zag laten helpen zwemmen

...that Jan saw Marie make the children help Piet swim

The point to note is that only the agreement between the first np and the first verb and the subcategorization restrictions between other nps and the final verb are to be encoded in the grammar. Hence it is possible to develop a CFG which can generate all these constructs. Bresnan et al [7] have shown this. However, such a CFG will not assign linguistically correct structural descriptions to these sentences. That is, it is possible to get a CFG that sufficient to handle Dutch in terms of its weak generative capacity but it has been shown that Dutch is not strongly context free. This time we have a real restriction, CFGs are not sufficient even in the technical sense as far as strong generative capacity is concerned.

1.2.5 Unbounded Branching

CFGs capture repetition through recursion. In many situations what we want is mere repetition and not recursion. For example, items in conjunction should really be all at the same level. Since there is no *a priori* bound on the number of items in conjunction, either we have to use recursive rules or make the number of rules potentially infinite. A sequence of 'a's can only be captured by one of the recursive rules

A $\rightarrow$ a A

or

A $\rightarrow$ A a

along with the base rule

A $\rightarrow$ a

or by the potentially infinite set of rules

A $\rightarrow$ a

A $\rightarrow$ a a

$A \rightarrow a\ a\ a$

and so on. Computational grammar formalisms require that the rule sets be finite and hence in general imposing nonexistent hierarchical structure through recursive rules is inevitable. CFG rules are suitable for dealing with recursion but not for handling repetition. It is worth noting that regular expressions, equivalent in computational complexity to the much simpler regular grammars, can directly and effectively deal with repetition.

In summary, CFGs impose *more structure* than really exists in some cases and *less structure* than required in others. CFGs impose linear structure when not appropriate, as in the case of relatively free word order languages. They impose too much of hierarchical structure in cases such as unbounded branching. As far as dependencies between different constituents are concerned, CFGs are unable to capture the required structural constraints effectively.

What then is good about CFGs? CFGs are excellent for handling situations where both linear and hierarchical aspects of structure are involved and nothing else is significant. If hierarchical structure is not involved, we may not even require CFGs and if linear structure is itself not significant, CFGs are no good. CFGs should not be applied in any situation where any kind of dependency between different constituents is involved. It should also be noted that going beyond CFGs and using more powerful phrase structure rules is not necessarily the ideal thing to do in all situations.

1.3 Current Status and Motivation

Despite the fact that several grammar formalisms have already been developed, the search for better formalisms continues. Theoretically, the available grammar formalisms are not fully satisfactory in explaining the human language faculty in all its ramifications. Computationally, the belief and hope that simpler and more efficient techniques exist, provides motivation for further search for better and better formalisms.

We will now make a brief survey of several important grammar formalisms that have made a significant impact on NLP, touching mainly those points that are relevant for our development of the UCSG system. More details can be obtained from the references we give. We will include in our survey Augmented Transition Networks (ATN), Lexical Functional Grammar (LFG), Generalized Phrase Structure Grammar (GPSG), Tree Adjoining Grammar (TAG) and the Paninian approach. We will also make a few general remarks about Government and Binding (GB) and other similar approaches.

1.3.1 Augmented Transition Network Grammar

Augmented Transition Networks were developed in the early 1970s around the same time that linguistics was also taking a new shape with Noam Chomsky leading the generative linguistics programme with his Transformational Grammar (TG). While the concepts behind ATN existed for some time before, it was Woods [53] who gave a formal shape to ATN. As is clear from his paper, ATN came up as an alternative to, and improvement over, Chomsky's TG. While claiming equivalence to TG in power, Woods argued that ATN offered a number of advantages including perspicuity, sufficient generative power, efficiency of representation, ability to capture regularities, efficiency of operation and flexibility for experimentation.

Marcus [31] has rightly observed that an ATN grammar can be viewed as a specification of a space of grammatical possibilities which says very little about how to search that space. While this division of knowledge into *what* and *how* is a boon to the grammar writer, ATN parsers are susceptible to becoming very inefficient. Unrestricted augmentation can make ATN as powerful as the Turing Machine. In general ATN is extremely powerful, much more powerful than needed, thus running the risk of becoming extremely inefficient. In this respect, ATN is not any better than TG.

ATN parses a sentence from left to right, simulating a nondeterministic machine either by using backtracking or through pseudo-parallelism. Nondeterminism appears in NLP because the information required to take the right decision may not be locally available at that point of time. Hence we will be left with only two options -

keeping track of all the possibilities simultaneously, or making an arbitrary guess and proceeding sequentially and backtracking to earlier choice points if forced later on. In both cases extra effort is expended in trying unfruitful paths. ATN blindly searches all the possibilities applying the conditions on the arcs to prune search. Since actions on one arc may affect the result of applying a condition on some other arc later, ATN grammars tend to become very complex and difficult to write.

ATN, like most other grammar formalisms, takes linear position of constituents in a sentence too seriously. The positional aspect is so much a part of the entire mechanism that it is not suitable for dealing with relatively free word order languages.

ATN handles long distance dependencies using its *hold mechanism*. Constituents which are obtained earlier than required are *held* in the hold register until a place is reached where that constituent is required but found to be missing. The ATN grammar never made any cognitive claims and was hardly known in linguistic circles. However, the hold mechanism seems to promise psychological reality also. See [23] for more on this.

While several experimental NLP systems have been built using ATN and the ATN mechanism is still being used effectively to solve limited parsing problems, ATN is no longer a serious candidate as a good grammar formalism for NLP. A good survey of ATN is [2].

1.3.2 Lexical Functional Grammar

Lexical Functional Grammar was developed in the early 1980s by Kaplan and Bresnan [29]. As the name suggests, LFG attaches a high degree of importance to the lexicon. While being sensitive to computational viability, LFG took a distinct linguistic flavour and has been making serious cognitive claims also. Thus LFG has come to be considered seriously in many schools of linguistics and, at least on some points, LFG is deemed to offer serious competition to the more recent theories within generative linguistics including the Government and Binding theory [26, 44].

There are parallels between LFG and ATN. LFG shares many of the merits and demerits of ATN. The CFG rules that generate the C-Structure of LFG are equivalent to the unaugmented part of the ATN. The Functional Descriptions of LFG parallel the augmentation in ATN. Thus in an abstract sense LFGs and ATN are equivalent and LFG grammars are as powerful as the ATN grammars, just too powerful.

LFG uses the less natural and less intuitive "double shafted arrow" mechanism for dealing with long distance dependencies. LFG chose this direct linking of functionally dependent constituents instead of step by step linkage through transitive chains of functional identifications under the pretext that this may require the introduction of otherwise unmotivated functions at intermediate F-Structure levels.

LFG uses the single operation of unification instead of the two separate operations of setting and checking the values of the feature dimensions as in ATN. This makes the LFG grammar somewhat easier to write. However, unification in LFG implies that a large number of feature structures are built and then rejected. Also unification itself is a costly operation.

LFG also takes linear position too seriously and thus fails to deal effectively with relatively free word order languages. See [4] for a discussion of LFG and Indian languages.

1.3.3 Generalized Phrase Structure Grammar

GPSG was designed to be a notational extension of CFG. GPSG is thus weakly equivalent to CFG and under many views of strong equivalence, it is even strongly equivalent to it. GPSG uses meta rules to accomplish many of the things that are handled by transformations in the TG framework. In GPSG a grammar is defined by giving a finite set of CFG rules and a finite set of meta rules which allow you to extend the list of CFG rules by the application of the meta rules. The meta rule concept is a very useful notational device and it can make the life of the grammar writer much simpler. There are restrictions on the application of meta rules and hence the overall system complexity is limited. Nevertheless, GPSG is essentially CFG and hence it carries with it all of the merits and demerits of CFGs except for providing notational convenience. See [19] for full details off GPSG. [18] presents the basic ideas underlying GPSG in a cleaner and clearer way.

1.3.4 Tree Adjoining Grammar

Tree Adjoining Grammars have been around since the mid 1970s. The significance of TAG is that it directly takes up the issue of the least powerful grammar that we need to handle natural languages. Taking for granted that CFGs are not sufficient, the proponents go on to show that TAG, which is only mildly context sensitive, is sufficient to deal with natural languages. TAG has been shown to be sufficient to handle both subcategorization dependencies and filler-gap dependencies. In fact TAG grammars can also deal with crossed dependencies which CFG cannot handle. See [27] for more details. TAGs permit polynomial time parsing with a worst case time complexity of $O(n^4)$, just n times worse than CFGs at their worst. While still not the ideal, this is a very significant improvement over the worst case complexity of TG, ATN, and LFG. While GPSG may claim a worst case time complexity of $O(n^3)$ based on its equivalence to CFG, the $\|G\|^2$ term within the constant factor, where $\|G\|$ represents the number of grammar rules, can become large enough to make it worse on the whole.

In other grammar formalisms, recursion in the CFG or equivalent component first builds up structures some of which get filtered out later when dependency constraints are applied. In TAG, however, dependencies are defined initially on bounded structures (trees) and recursion simply preserves them. This leads to perhaps one of the most significant aspects of TAG namely its elegant way of handling long distance dependencies. In a sense there are no long distance dependencies at all since all such dependencies originate within bounded structures which only get expanded later.

TAGs use ordered trees and thus have the same kinds of problems as other grammar formalisms we have seen above when it comes to handling free word order languages. With the more recent work in TAG however, [50, 40, 41] the new quasi-trees are coming closer to the karaka charts of the Paninian grammar.

1.3.5 Paninian Grammar

We will now take up the Paninian approach as developed by the NLP group at I.I.T., Kanpur, India [4]. The Paninian Grammar has been specifically designed for elegantly and efficiently dealing with Indian languages which are characterized by relatively free word order and a rich system of morphological inflections and/or postpositions. Here fundamental ideas from Panini's grammar of Sanskrit and a number of other ancient Sanskrit works have been collected, adapted and reinterpreted for NLP work in modern Indian languages.

In this theory, syntactic analysis mainly concerns itself with the assignment of functional roles, here called *karaka roles*, to the various constituents in a given sentence. Thus this theory is similar in its spirit to the Case Grammars as given by Fillmore [16, 17] and many others who followed in the west. In a way, Fillmore's case grammar is a rediscovery of what Panini and other ancient Indian grammarians had already said long before.

Paninian theory seeks to find out where the information required for assigning the karaka roles comes from. In Indian languages, this information is encoded primarily in the surface case markers, here called *vibhakti*. Paninian approach treats morphological inflections as well as postpositions as indicators of vibhakti. Linear position, which is all important in western theories, is given only a secondary importance. In Indian languages position encodes only secondary information such as focus and topicalization. The mapping from vibhakti to karaka roles is not always simple and straightforward, and the core of the theory deals with the design of grammars and parsers to achieve this mapping efficiently. Assignment of karaka roles is basically a matter of constraint satisfaction - any set of assignments that satisfy all the constraints simultaneously is a valid syntactic description of the given sentence.

The Paninian grammar has been shown to be simple, computationally viable and efficient, yet powerful enough to deal with the full range of syntax of Indian languages. More recently, an extended notion of vibhakti, that views linear position as also a kind of surface case marker, has also been contemplated to enhance the scope of the approach to include English and other positional languages [5, 3]. However, the

Paninian approach was conceived and designed especially for Indian languages and extensibility to other kinds of languages is only an afterthought. Paninian grammar neglects hierarchical structure and is therefore likely to be very inefficient for languages like English which show rich hierarchical structure. Even for Indian languages Paninian grammar is less efficient than possible as we will see later.

1.3.6 Government and Binding Theory

Finally, we need to make a few observations about Government and Binding and other theories that linguists within the modern generative school have been proposing. Firstly, these theories have quite different aims. They are not theories of how humans understand what they hear or how they synthesize natural language sentences to convey whatever they want to. Here syntax is not viewed as a step towards a bigger goal, syntax is rather viewed independently with a deplorable apathy towards its relation to meaning. These theories are almost exclusively limited to *explaining* the native speakers' intuitions about grammaticality of utterances. While NLP theories may also restrict their scope to syntax, they admittedly work within the scope of bigger theories where meaning is extremely important. This difference is fundamental and has far reaching effects on the nature and appearance of the entire theories, making comparisons very difficult if not impossible.

Secondly, and as a consequence of the above, these theories do not subscribe to a process model. Syntactic analysis is not delineated in terms of inputs, outputs, and step by step actions. There can be no questions about what is done first or last, what thing takes what amount of time, memory or any such resource. Thus these theories are not computational theories at all. While computational models can be derived from such theories, a whole lot of very significant decisions will have to be made in this derivation process and the resulting model cannot be taken to depict the original theory as proposed.

For example, if we have to take the NLU view of GB, we will have to perhaps take the PF-Structure as input and produce the LF as output. What GB does is to start from D-Structure and reach both PF and LF through the S-Structure. This requires reversal of the path between PF and S-Structure. Does the PF retain enough information to reconstruct the S-Structure? What is the role of the D-Structure then? Both PF, the input to an NLU model and LF, the final output that we are looking for, get only a peripheral treatment in most works on GB.

It is significant that none of the theories we have discussed above have any component that directly parallels the D-Structure. The main thing that happens between D-Structure and S-Structure is *move-alpha* - move any constituent from anywhere to anywhere, subject to all the constraints scattered in different modules of the GB system. A computational model cannot afford to try to move all existing constituents to all possible positions and the order in which the various constraints are checked cannot be left completely unspecified. As we have already observed earlier, this purely specification oriented view of grammar is hopeless as a computational model.

Finally, like all other western theories, ordered trees are at the heart of GB theory raising once again the same questions about suitability for relatively free word order languages. GB defines the subject of a sentence as the np immediately dominated by s. In Telugu, there can be several nps, not just one np immediately dominated by s. Which of these is the subject? GB theorists blame NLP researchers for not taking the *normal* or *unmarked* word order seriously. It should be noted that in relatively free word order languages, the distinction between normal word order and other word orders is in terms of phenomena such as focus and topicalization which are of secondary importance in NLP. The most important aspect is the gross meaning of the sentence as depicted in the functional structure, which is invariant across different possible word orders.

As a consequence of all this, we do not find it meaningful to make comparisons with GB and similar theories of linguists. There is a very rich set of references for GB. Two of perhaps the most important are [12, 13]. Peter Sells [44] gives a brief and comparatively clear introduction to GB. There is also no dearth of critiques on generative linguistics, on GB and even on Noam Chomsky and his views in particular. Amorey Gethin [21] is a very good criticism of the modern generative linguistics.

In summary, while different grammar formalisms have their own strengths and weaknesses, none of them seem to be fully satisfactory. The search for better grammar formalisms is fully justified. Keeping the positive and negative aspects of these grammar formalisms on the one hand and the requirements of a good grammar formalism on the other hand, we have attempted to find a better alternative which we have called UCSG.

1.4 Essence of UCSG

Our primary aim has been to develop a computationally viable grammar formalism that leads to simple grammars and efficient parsing for both positional languages and relatively free word order languages.

In UCSG we divide and conquer. By appropriately dividing the problem into subproblems we get to see where exactly relatively free word order languages differ from positional languages and what is really common between the two. We apply the least complex grammars that are sufficient for each of the subproblems thereby leading to very efficient parsing. Also, grammars for the subproblems can be written separately and grammars become easy to write.

Grammars can be viewed as sets of constraints on the structure of sentences. There are constraints on linear position of constituents, hierarchical nesting of constituents one inside the other and functional dependencies between different constituents in a sentence. We find that these three primary kinds of structure inherent in human languages, namely linear, hierarchical and functional structures, lend themselves naturally for analysis by three independent modules. The three modules in UCSG are called linear (L), hierarchical (H) and functional (F) components. Both the grammar and the parser are divided into L, H and F parts. Correspondingly, there will be three levels of syntactic representation called L-Structure, H-Structure and F-Structure. While other grammar formalisms also tackle linear, hierarchical and functional aspects some way or the other, division of labour into these three separate modules, especially the introduction of an independent H level is the highlight of the UCSG formalism.

F-Structure depicts the assignment of functional roles to the various participants in a predication. The F part of the grammar provides constraints for role assignment. Functional roles should be so selected that role assignments can be made based mainly on syntactic information. We propose a set of eight functional roles motivated by the question-answer point of view. We employ a combination of top-down and bottom-up strategies for role assignment. The subcategorization frames associated with verbs provide the top-down expectations and the surface case markers associated with the nouns provide the bottom-up information. Any set of role assignments that satisfy all the F constraints simultaneously gives a valid F-Structure.

Each clause in a sentence corresponds to one atomic predication. It is a general principle of syntax that participants of a clause do not cross the clause boundaries. Knowing the clause structure therefore greatly helps the functional structure analysis since functional structure analysis can then be localized to the respective clauses. The F part of the grammar can be specified for a single clause only and the grammar becomes simpler and easier to write. Functional role assignment is essentially local to a clause. In UCSG we obtain the hierarchical structure of clauses and the clause boundaries before we attempt functional structure analysis. We show that clause structure can be determined before and without applying the functional structure constraints. H-Structure analysis logically precedes F-Structure analysis in UCSG. An independent H level analysis that is done before and without the aid of F level constraints is the highlight of UCSG.

We observe that there are strong constraints on the sequences of verbs and certain clause boundary markers called sentinels. We exploit these constraints to identify clauses, to determine the hierarchical nesting of clauses and even to determine the clause boundaries although only partially. H-Structure depicts this hierarchical structure of clauses. The H grammars for different languages show only parametric variations even across positional and relatively free word order languages. The grammar of clause structure is highly language invariant. Hence the name Universal Clause Structure Grammar.

We observe that the atomic units at both hierarchical structure and functional structure level are groups of words rather than individual words. We therefore propose that all potential word groups be identified first and both hierarchical structure analysis and functional structure analysis be carried out on sequences of word groups rather than on sequences of words. This reduces the effective length of the sentence and increases the efficiency of parsing. The L part of the grammar provides constraints for group formation. We show that the L grammar constraints in both positional and relatively free word order languages are primarily constraints of linear position. Hence the name linear structure.

The only way functional dependencies can span across several clauses is through the nesting of clauses. These dependencies have got something to do with the hierarchical structure of clauses but they have nothing to do whatever with *distance* - long or short. The concept of linear distance is irrelevant here. We get rid of the problems of long distance dependencies by working from whole to part instead of working from left to right. We determine the clause structure first and then analyze the functional structure clause by clause. We start from the main clause and go recursively into the nested clauses. We pass on information and expectations about displaced, shared and missing participants down the hierarchy of clauses. This additional information makes functional structure analysis of a clause practically completely independent of the functional structure of any other clause in the sentence. There will no longer be any long distance dependencies. Thus long distance dependencies, which require special mechanisms in other grammar formalisms, are handled in a very simple and elegant way in UCSG.

In UCSG we thus work from whole to part rather than one end to the other. We know the hierarchical structure of clauses and clause boundaries before we attempt to analyze the functional structure of individual clauses. In all other grammar formalisms sentences are processed left to right. At any point in this process, decisions are taken based only on the information gathered from the part to the left, the part that has already been seen. We have really no idea of what lies ahead but for certain expectations which can be built based on the things we have already seen. Working from whole to part has the advantage that we always have a global picture of what we are doing before we fill up the minor details.

We show that all potential word groups in a sentence can be obtained using finite state machine power in linear time. We also show that hierarchical structure analysis can be done using a very small number of context free grammar rules in cubic time - cubic in the number of verb groups and sentinels, which is typically much smaller than the number of words in the sentence. We further argue that hierarchical structure analysis can also be done in linear time if we bring to bear psycholinguistic and statistical evidence on the limitations of hierarchical structure of natural language sentences. Finally, we show how the localization of functional structure analysis to the local domains of the respective clauses makes parsing in UCSG highly efficient on the whole.

In conclusion we see that the UCSG formalism has several merits. Firstly, we see that the division of work into three independent modules helps to discover, and hence exploit, what is common between a relatively free word order language like Telugu and a positional language like English. UCSG is equally well suited for parsing positional languages and relatively free word order languages. In fact we will see that even cross serial dependencies as in Dutch can be handled. Secondly, we get an elegant solution to the problem of long distance dependencies. Thirdly, grammars can be written more easily than with other grammar formalisms. Fourthly, we note that much more efficient parsers can be written than with ATN, LFG, GPSG, TAG and the Paninian approach. Lastly, and most importantly, the division of labour into three modules that have a very close correspondence with the three aspects of structure of human languages, will hopefully offer new and useful insights into the nature of our languages.

Prototype parsers for English and Telugu have been developed and tested [36, 33]. Sample inputs and outputs from the English and Telugu parsers are given below. More details of these examples can be found in the next chapter.

25) The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night.

26)

baaMkunu doocaalanukonna doMgalu raatri aDavilooniki paaripooyi uMTaarani pooliisulu nammaaru

Notes:

See parse outputs in the following pages.

1. The numbers at the top indicate the serial number of the sentence being parsed, the parse number, and the clause number. Clauses are numbered. Main clause is always numbered one.

2. Embedded clauses are shown with the corresponding numbers in the roles in which they appear. Sentinels which mark the clause boundaries and relate clauses to one another are depicted in the clause-link (CLS-LINK) role.

3. The complete sentence is given at the bottom for reference. The part of the sentence that goes into the current clause is shown at the top. The linear order of words and embedded clauses is retained and shown here.

4. In clauses 3 and 4 in this example, the subject 'the dacoits' is borrowed from outer clauses and shown blinking on the screen.

UCSP−English :

1 : 1 : 1 : The police believed 2

SUBJ

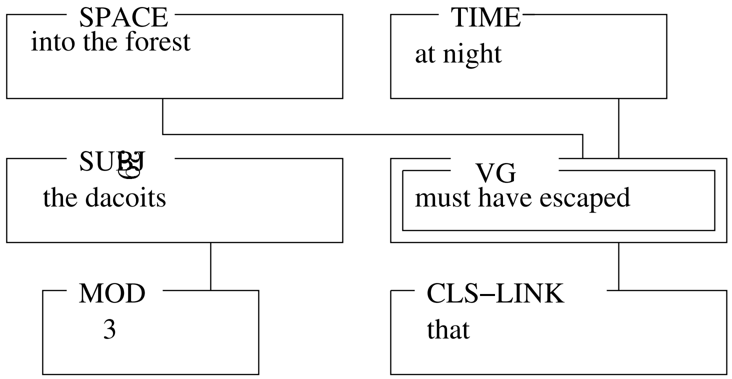
VG
believed

OBJ
2

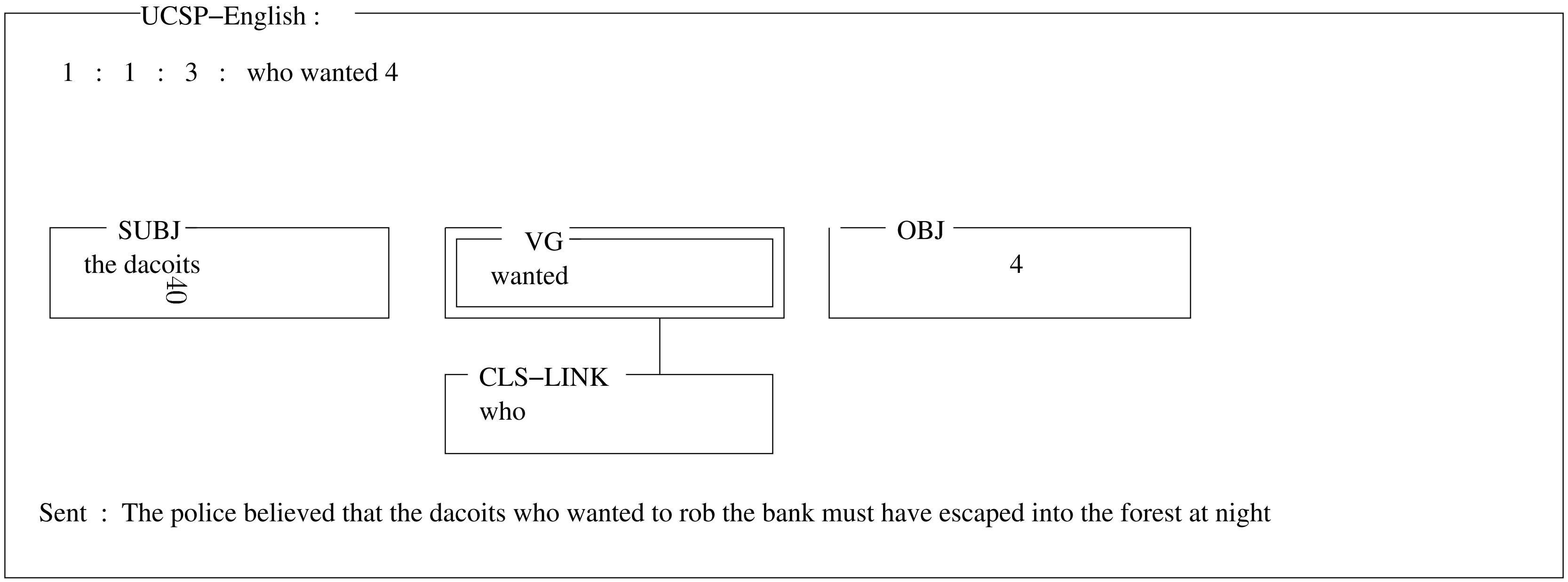
Sent : The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night

UCSP–English :

1 : 1 : 2 : that the dacoits 3 must have escaped into the forest at night



Sent : The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night


UCSP–English :
1 : 1 : 3 : who wanted 4
SUBJ
the dacoits
VG
wanted
OBJ
4
CLS–LINK
who
Sent : The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night

UCSP−English :

1 : 1 : 4 : to rob the bank

SUBJ
the dacoits

VG
to rob

OBJ
the bank

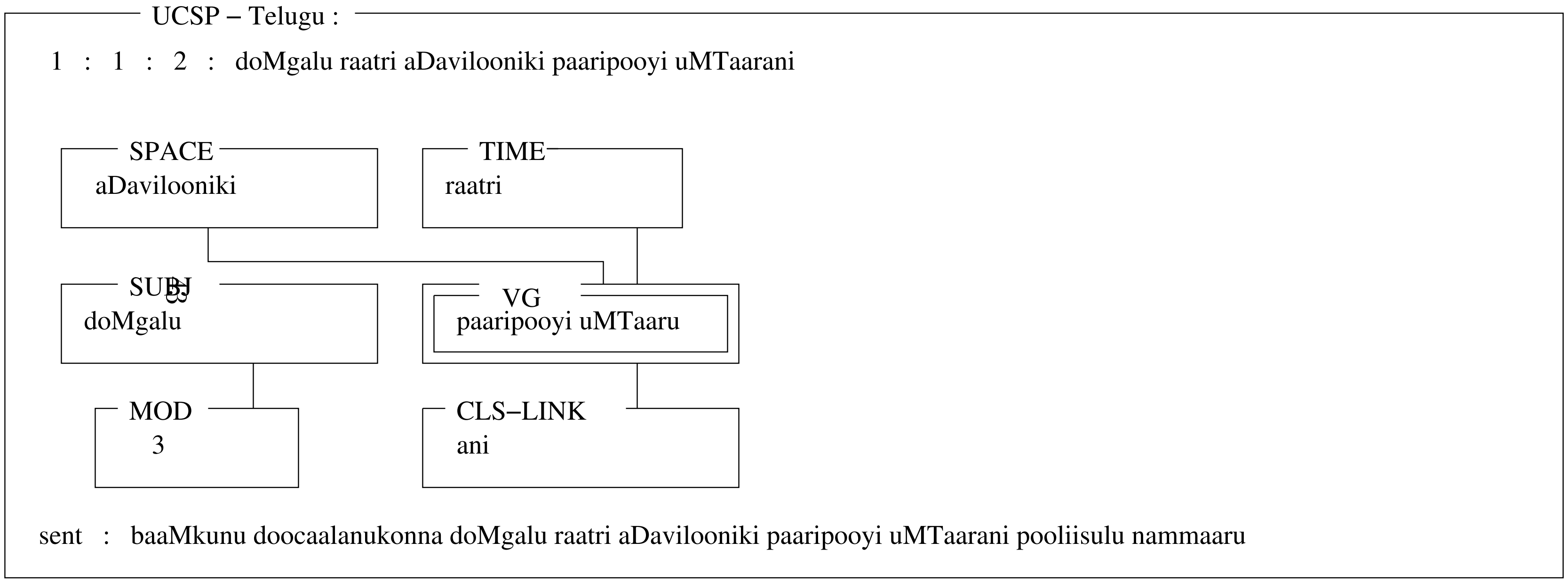
Sent : The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night

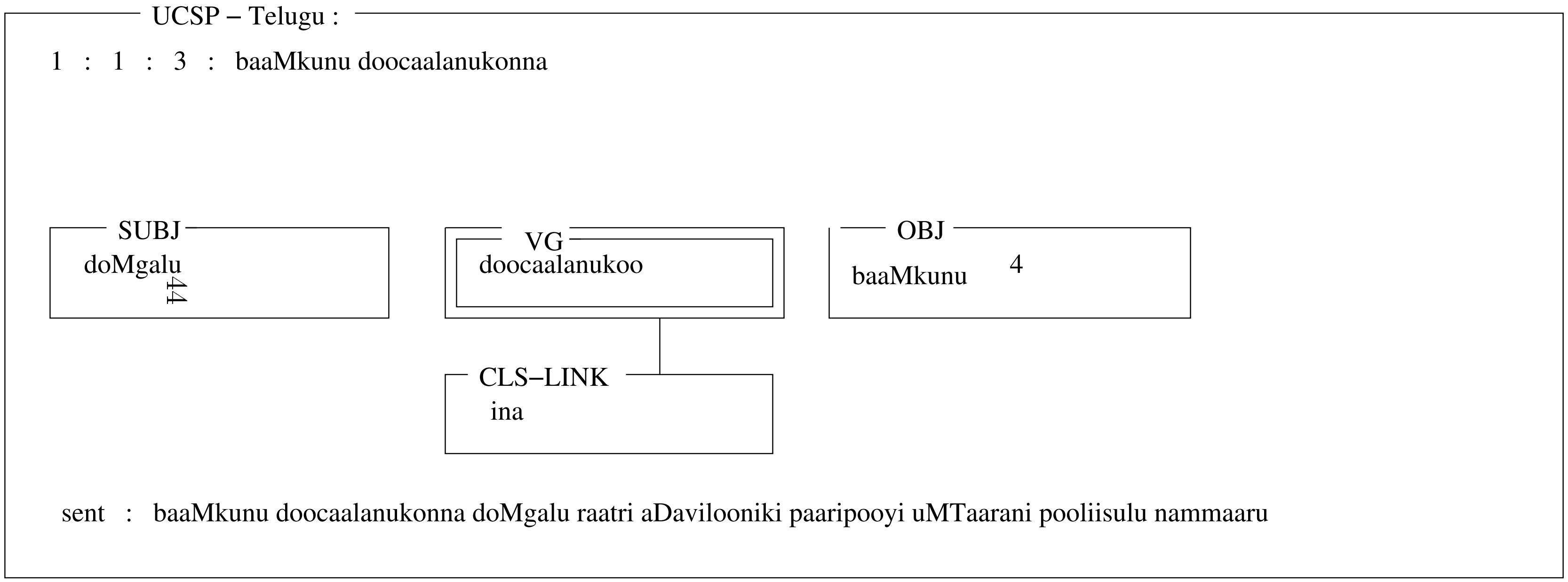
UCSP – Telugu :

1 : 1 : 1 : pooliisulu nammaaru

SUBJ	VG	OBJ
pooliisulu	nammaaru	2

sent : baaMkunu doocaalanukonna doMgalu raatri aDavilooniki paaripooyi uMTaarani pooliisulu nammaru


UCSP – Telugu :
1 : 1 : 2 : doMgalu raatri aDavilooniki paaripooyi uMTaarani
SPACE
aDavilooniki
TIME
raatri
SUBJ
doMgalu
VG
paaripooyi uMTaaru
MOD
3
CLS–LINK
ani
sent : baaMkunu doocaalanukonna doMgalu raatri aDavilooniki paaripooyi uMTaarani pooliisulu nammaaru


UCSP – Telugu :
1 : 1 : 3 : baaMkunu doocaalanukonna
SUBJ
doMgalu
44
VG
doocaalanukoo
OBJ
baaMkunu
4
CLS–LINK
ina
sent : baaMkunu doocaalanukonna doMgalu raatri aDavilooniki paaripooyi uMTaarani pooliisulu nammaaru

1.5 Thesis Organization

In chapter 2 we develop our UCSG grammar formalism in detail. We start chapter 2 with our aims and the definition of our problem. We discuss the UCSG view of syntactic analysis and then get into the details of linear, hierarchical and functional structure analysis. Chapter 3 describes the implemented English and Telugu parsers. We end with conclusions and pointers for further work in chapter 4.

Chapter 2

Universal Clause Structure Grammar

In this chapter we will develop our grammar formalism - Universal Clause Structure Grammar (UCSG) in detail. We define our problem in section 2.1. In section 2.2 we will give the basic approach taken by UCSG. In the following three sections we will discuss the three main modules in UCSG. Details of an implemented parser are given in the next chapter.

Let us first recapitulate some of the points we mentioned in the first chapter so that we can fix our aims and scope clearly. The research work presented in this thesis is about Natural Language Processing (NLP), in particular, about Natural Language Understanding (NLU). In NLU written sentences are taken as input and formal, symbolic representations of the meaning of the utterances are produced as output. The process of mapping from the inputs to the outputs is termed 'the process of understanding', or simply 'understanding'.

Mapping a given utterance to its meaning representation is a very complex process. Languages exhibit complex structure and a detailed and systematic analysis of the structure of natural language sentences is invaluable in determining their meaning. Syntax deals with the problem of analyzing the structure of natural language sentences and producing structural descriptions. These structural descriptions can then be analyzed further by semantic and pragmatic components to obtain the meaning and the intentions behind the sentence. The scope of this research is limited to syntactic analysis. We develop a theory of syntactic analysis of natural language sentences. We take the linguistic approach and develop a grammar formalism called UCSG. We deal with the structure of natural language sentences, grammars that specify these structures and the parsing algorithms that apply the grammar on given sentences and produce structural descriptions.

Grammar of a language is a formal specification of the valid structures in that language. Since our focus is on syntactic analysis, we use the term grammar in this limited syntactic sense. A grammar formalism is a meta language that specifies the structure and hence the properties of grammars written in that formalism. We consider a grammar formalism to be good if simple, elegant and universal grammars can be developed, exhaustive computational grammars can be written with ease for different human languages, efficient parsers can be developed within the formalism and the syntactic descriptions produced by these grammars clearly depict linear, hierarchical and functional structure inherent in the given sentence.

It must be emphasized that from an NLP point of view the theory should specify what structural descriptions are produced for given sentences and how. The input is a sentence in natural language and the output is a symbolic representation of the structure of that sentence. The orientation in some of the syntactic theories developed within linguistics is completely different. Some of these theories are not computational theories at all.

Most grammar formalisms developed in the west have an obsession for linear precedence constraints. This not only makes it difficult to deal effectively with relatively free word order languages but also introduces spurious problems such as the problem of long distance dependencies. While the Paninian grammar has been shown to be both simpler and more efficient for dealing with Indian languages, which belong to the class of relatively free word order languages, it neglects hierarchical structure, an extremely important aspect of syntax. This makes the Paninian grammar highly inefficient for positional languages like English and less efficient than possible even for Indian languages.

We are interested in theories of syntax that discern generalities across human languages. In particular we require that both positional languages and relatively free word order languages be handled efficiently and as uniformly as possible. Also, emphasis should be placed on efficiently and elegantly handling the run-of-the-mill phenomena rather than obscure, infrequent and idiosyncratic examples from specific languages.

The aim is to develop the least powerful grammar that is sufficient to deal with the syntax of natural languages. Computational complexity and power of grammars are directly related. The simplest grammars are the most efficient. Other grammar formalisms have followed the general approach of going for a more powerful type of grammar if a simpler grammar is found to be inadequate for dealing with the whole of syntax. We believe that it is instead wiser to ask which aspects of syntax really require grammars of which power. By dividing the task carefully and applying the right kind of grammar for each part, we can obtain much simpler grammars as well as much more efficient parsers on the whole. Our main concern in UCSG has been how to divide the problem of syntactic analysis so that we get a simple, universal grammar and an efficient parser. Keeping the desirable qualities of good grammar formalisms on the one hand and the strengths and weaknesses of available grammar formalisms on the other, we have endeavoured to develop our own grammar formalism called UCSG. We will first give our problem definition and then proceed to see the UCSG approach to syntactic analysis.

2.1 Problem Definition

We aim at developing a grammar formalism with the following properties. It may be noted that while some of these properties can be quantified and measured, others are more qualitative and judgemental in nature.

- *Computational Viability* : Grammars written within the grammar formalism must be amenable to automatic processing by a computer. Computational grammars need to be written exhaustively, in full detail and so precisely that a machine can mechanically apply the grammar and carry out syntactic analysis. There cannot be any parts that require human intuitions, commonsense or on-the-spot judgemental decision making. We also wish to depend mostly on syntactic information, resorting to semantic and pragmatic information only when really necessary.

- *Universality* : The grammar formalism must discern and focus on the universal features of human languages rather than on idiosyncratic features of individual languages or language families. In particular, we require that positional languages like English and relatively free word order languages like the Indian languages be handled in as uniform a way as possible.

- *Simplicity and Ease of Grammar Writing* : The grammars written within the grammar formalism must be simple, neat and elegant. It should be easy to write exhaustive computational grammars for different human languages.

- *Structural Descriptions* : The structural descriptions produced by the parser must clearly depict linear, hierarchical and functional structure of the input sentences.

- *Efficiency* : Parsing should be efficient. The grammar formalism should not only enable computational grammars to be written but also provide a basis for determining how exactly to apply the grammar rules and constraints for efficient parsing.

- *Number of Parses* : Some parsers generate hundreds of parses for a given sentence, only a few of which may be correct. Picking out the appropriate parses would be so very difficult that such parsers are practically unusable. We require our parser to generate only a few parses.

2.2 The UCSG Approach to Syntactic Analysis

To understand some object completely, we need to understand three kinds of things. Firstly, we need to know the form and function of the object as a whole. Secondly, we need to know the form and functions of all its parts, recursively going down to subparts and subsubparts in the hierarchy. Thirdly the interrelationships and functional dependencies between the parts or subparts must also be understood.

This approach is very general and we believe that it can be applied fruitfully for the study of any kind of object. In this thesis we will apply this approach for the analysis of natural language sentences. In natural language sentences, form and function correspond to linguistic structure and intended meaning respectively. We need to break up the given sentence into parts and subparts and depict the structure of the sentence in terms of these parts and subparts, their internal structures, functions and interdependencies, so that we can finally understand the structure and meaning of the sentence as a whole.

In this thesis we are dealing with syntax and hence we are primarily concerned only with structure and not with meaning. Within a syntactic framework all that we aim to do is to obtain a structural description of the given input sentence. We are therefore concerned here with the identification of suitable syntactic units and then with describing a given sentence in terms of appropriate structures of such units.

However, in NLP we view syntax as a step towards the final goal of understanding the meaning. Consequently, structure and meaning cannot be completely divorced from one another. Form follows function. The division of the sentence into parts and subparts as well as the interdependencies between these parts and subparts are dictated by concerns of semantics. We should break the sentence into parts and subparts so that these structural units help us to identify meaning units rather than distort or obscure meaning. Structural units must correspond to meaning units as closely as possible.

2.2.1 Predications and Functional Structure

What then are the meaningful structural units in a sentence? How do we describe the structure of a sentence in terms of these units? At one level, the entire sentence is a meaningful unit. It is an independent utterance in its own right, intended to make sense to the listener. But consider

27) I went to the bank. I withdrew some money.

28) I went to the bank and I withdrew some money.

29) I went to the bank to withdraw some money.

In 27) we have two sentences while in 28) and 29) we have a single sentence. In all cases however, two related but independent events are described - going to bank and withdrawing money. This indicates that there must be another level of description, possibly smaller than a sentence such that the structural units at this level make sense.

People use language for communicating with fellow human beings and at one level the units of information conveyed can be taken as assertions about states, actions, events, descriptions of objects, commands to perform an action and questions about such actions, events or descriptions. Each such unit describing one action, event or state can be called a *predication*. We will use the same term 'predication' whether we are making an assertion, asking a question or making a request. Predications are meaningful units. It is better to work with predications than with entire sentences, which may involve one predication or several.

Also, predications are minimal units in the sense that parts of a single predication do not always make complete sense. While optional modifiers may be removed from a predication without introducing incompleteness, these optional parts also belong to the predicate, they do add to the meaning of the predication. Also, sentence fragments are not independent units of meaning. Consider

30) Q: Where had you been?

A: To the Library.

31) Q: What do you like most?

A: Playing Hamsadhwani on the flute.

These fragmental replies represent incomplete predications that can be understood only in the discourse context in which they are set. We need to be able to discern that these fragments on their own are incomplete. Analyzing sentence fragments is really a discourse level problem.

Every predication specifies a complete action, an event or a state. There may be several participants in the whole predication - the doers and the done-tos, the place, the time, the reason, etc. We believe that a good structural description must clearly identify such functions and relate them to the structural parts of the sentence. We believe that identification of such functions at a grammatical level of abstraction is both possible and highly desirable. Thus the English sentence

32) I bought a computer

can be ascribed by the predicate-argument structure : 'buy(I,computer)' with the convention that the first argument of the predicate 'buy' specifies who bought and the second argument specifies what is bought. We use a default ordering of arguments so that we can avoid specifying each time what the arguments mean. As a consequence, the order of arguments in such a predicate-argument structure is important. By definition, arguments are essential, without them the predication would not make complete sense. A predicate-argument structure normally depicts only mandatory participants. Apart from these essential participants, a predication may involve other optional participants such as place and time modifiers. Optional participants also contribute to meaning. Can't we ask questions about place or time of some predication and expect meaningful replies? All the participants, whether mandatory or optional, are part of the predication.

We will call this aspect of structural description which depicts the functions or roles played by the various constituents in a predication as functional structure or F-Structure for brevity. An F-Structure shows all the participants in the predication along with the functional roles they play. Both mandatory and optional roles are depicted.

2.2.2 Clauses and Hierarchical Structure

Structurally, each predication in a natural language sentence manifests itself as a clause. Hence clauses form an appropriate level of structural description. A simple sentence makes a single predication and hence it has a single clause. A complex sentence makes one main predication but some of the participants of this predication may themselves be other predications each of which appears as one clause in the sentence. Clauses can be nested one inside the other in several ways and over several levels. A clause can be used as the 'subject' or 'object' of another clause or as a modifier of place, time, etc. Clauses can also be used to modify nouns. See examples below.

33) Computers which include the new chip are not reliable
34) We all know who makes these chips
35) That the company kept the secret for so long annoyed everyone
36) We did not expect that this company would do such a thing
37) That the computer which I bought from the well-known company was defective has been annoying me
38) After the bug in the new chip was announced, the company's sales have dropped sharply
39) The company's sales have dropped sharply because it cheated its customers.

A good structural description must separate out the various clauses in a sentence and identify the hierarchical relationships between these clauses. We denote this aspect of structure by the term hierarchical structure or H-Structure for short.

It is a general principle of syntax that clauses show well defined clause boundaries and the participants of a clause do not normally cross the boundaries of that clause. This is true of all languages, both positional languages and relatively free word order languages. Hence it is highly desirable that the hierarchical structure of clauses be determined first. With the clause boundaries known, the task of assigning functional roles to various participants can be localized to the respective clauses. This makes the grammar very simple and the parser extremely efficient compared to carrying out functional structure analysis of the entire sentence at a stretch. This is the crux of UCSG.

In UCSG we show that it is possible to determine clause structure before and without applying the functional structure constraints. We will show that we can carry out hierarchical structure analysis with CFG power. We will not only be able to identify the clauses and determine their nested structure but also obtain the clause boundaries partially.

Hierarchical structure of clauses is the most crucial aspect of UCSG. We show that analyzing the hierarchical structure before analyzing the functional structure offers simple and elegant solution to the problem of long distance dependencies. Hierarchical structure grammar happens to be the most universal aspect of grammar. The hierarchical structure grammar of relatively free word order languages is identical to that of positional languages but for parametric variations. Hence the name Universal Clause Structure Grammar.

2.2.3 Word Groups and Linear Structure

At another level, words themselves can be meaningful units. After all we use words to talk about objects, their properties, actions, states, etc. However words are not the ideal units of syntactic description. Firstly, not all words in a language have independent meaning. Consider for example the words 'and' and 'the'. Secondly, Even words which have independent meaning often lose or change their intrinsic meaning when they combine with other words in a sentence. The primary meaning of 'keep' is lost in 'keep thinking'. Instead the word 'keep' here imparts the sense of repetition and habituality to 'thinking'. 'keep thinking' is one meaningful unit and it cannot be broken up further without loss of meaning. For the same reason, 'has been thinking', 'is about to think', 'had to think' and 'the red book' should all be treated as atomic units.

Such groups occur in both positional languages and relatively free word order languages. Thus 'will keep on going' of English and its nearest equivalents ('jaatee rahte haiN') of Hindi and ('pootuu uMTaaru') of Telugu are all single groups. Word groups often translate as a whole from one language to another. A group of several words in one language often has a single word equivalent in another language. Whole groups can be given out as fragmental replies to questions. These groups act as atomic units in filling up the functional roles in a sentence. In

40) My computer keeps crashing

the 'subject' is indicated by the group of words 'my computer' and the action is indicated by the group 'keeps crashing'. At the level of taking up the participant roles in a predication, none of the individual words in this sentence can stand on their own. Groups act as atomic units throughout syntax. Groups are the minimal meaningful units of structure, not words. Syntactic theories which deal directly with words rather than with word groups are more complex than need be. They are also insensitive to meaning.

We define a *word group* or simply a *group*, as a structural unit that can as a whole play one role in some predication. Typically groups are contiguous sequences of words and hence the name. We will use the term 'group' rather than the more usual term 'phrase' to avoid confusion with different uses of the term 'phrase'. When linguists say that a sentence consists of a 'noun phrase' and a 'verb phrase', for example, their use of the term 'verb phrase' implies not just the auxiliary and main verb sequence but the whole of the sentence except for that particular 'noun phrase' which has been explicitly mentioned. Their 'verb phrase' may in fact include one or more 'noun phrases'. The usual 'phrases' used in syntactic theories do not necessarily correspond to meaningful units at any one specific level. Diving a sentence into an np and a vp is like dividing a human body into a hand and the rest of the body. The parts are neither of the same type nor roughly equal in size. It is well known in computer science that when we divide the whole into parts, the parts should be of the same type and more or less of the same size to gain any real advantage of the divide and conquer strategy.

It is important to note that the linear order of words within a word group is always significant. Changing the order of words makes the group meaningless or at least changes or distorts the meaning. This is true in both positional and relatively free word order languages. We will argue that hierarchical and functional structure within word groups, if any, cannot in general be captured within a syntactic framework. We will show that linear order constraints are both essential and sufficient for recognizing word groups. The 'linear structure' of a sentence depicts this linear ordering of the words within word groups. We will also show that this linear structure analysis can be done with finite state machine power in linear time.

While there is considerable freedom in the order of word groups in relatively free word order languages, linear order of word groups plays a very important role in the functional structure of positional languages. Linear order constraints are, however, just one of a variety of types of constraints which different languages employ for specifying the functional structure in a clause. We view linear position, morphological inflections, prepositions and postpositions all as different manifestations of one and the same phenomenon which we shall call 'surface case marking'. We believe that the primary purpose of all these forms of surface case marking is the same - all of them are indicators of functional roles that various participants may take. All these varieties of constraints help us, to different degrees in different classes of languages, to assign functional roles to the word groups. Thus linear position of word groups within a clause is a matter of concern only in the functional structure analysis. We will not therefore include this aspect within our definition of 'linear structure'.

In UCSG we use groups and clauses as the primary units of syntax. Having decided the units of structural description and the nature of structural descriptions we would like to produce, let us see how exactly we carry out syntactic analysis of sentences.

2.2.4 UCSG = L + H + F

Our primary concern in developing UCSG has been how to divide the problem of syntactic analysis so that we get a simple grammar and a highly efficient parser. In UCSG we ask which aspects of syntax really require which types of grammar and break the whole problem into subproblems accordingly. We divide and conquer. Each module employs the least powerful type of grammar that is sufficient for that subproblem. Hence the overall system becomes very efficient. Other merits of UCSG follow from the way we divide the problem.

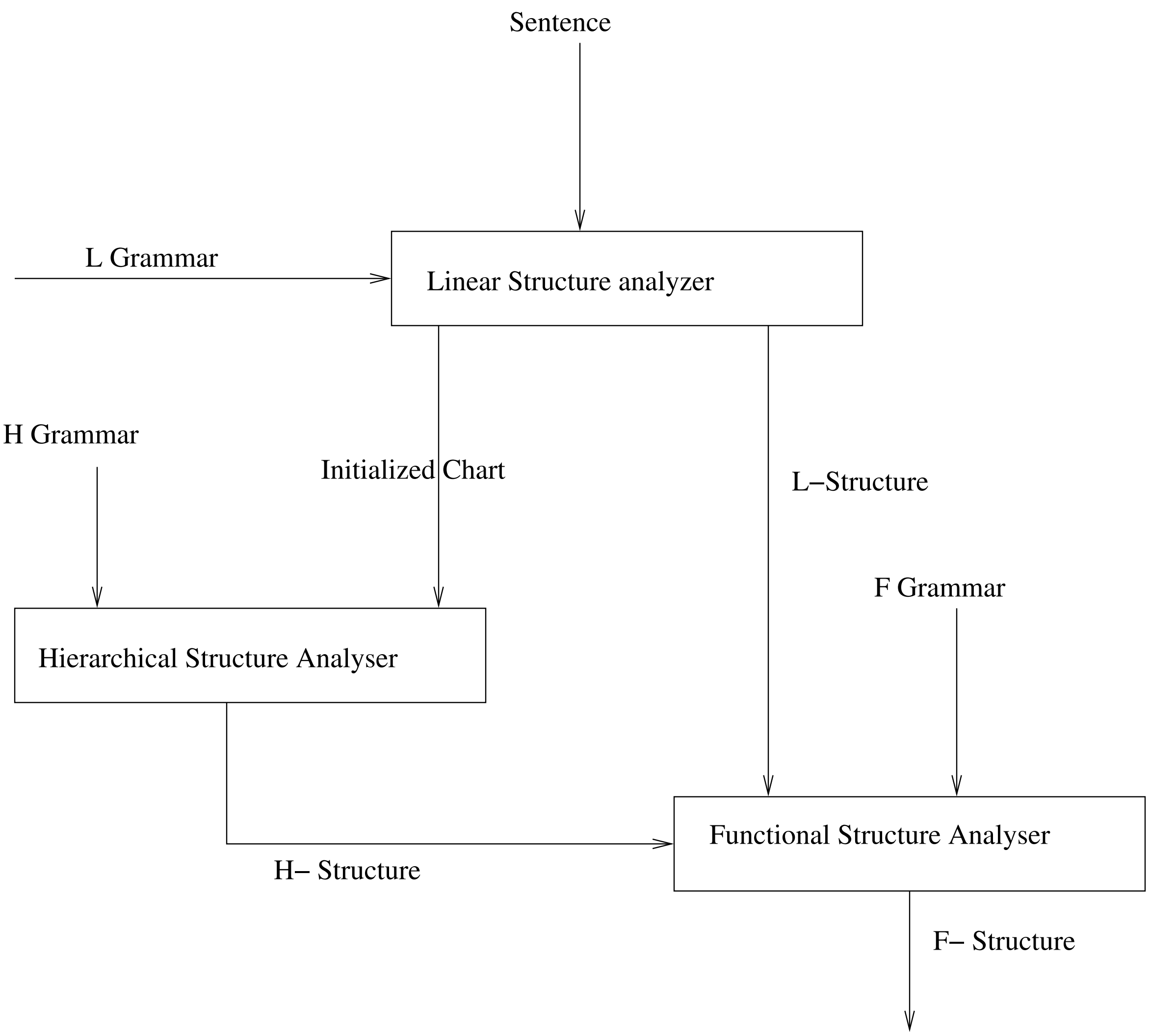
We have shown above that the three primary aspects of structure of natural language sentences are the linear, hierarchical and functional structures. We have found that these three primary kinds of structure inherent in human languages lend themselves naturally and elegantly for analysis by three independent modules. In UCSG both the grammar and the parser are divided into three modules respectively called linear (L), hierarchical (H) and functional (F) modules. While other grammar formalisms also tackle linear, hierarchical and functional aspects some way or the other, division of labour into these three separate modules, especially the introduction of an independent H level is the highlight of the UCSG formalism.

We take word groups and clauses as suitable units of structural description. These units have internal and external requirements. The requirements of internal structure are useful in identifying the units and the external requirements are useful in assigning appropriate roles to these units. Locally, there may be several different possible internal structures for a unit and a particular unit can have more than one possible role to play. Constraints of the grammar taken together determine the structural descriptions which are globally consistent within a given sentence.

The basic building blocks for determining both hierarchical structure and functional structure are the groups. In UCSG we therefore start off with words and group them into word groups using the constraints of linear structure. We then take up the hierarchical structure analysis. Having determined the clauses, their hierarchical relationships and the clause boundaries, we carry out functional structure analysis essentially in a clause by clause fashion. We bring to bear the top down constraints of subcategorization frames and the bottom up constraints of surface case marking. A set of role assignments that simultaneously satisfies all these constraints forms a valid functional structure. We start with the main clause. If any of the roles in this clause happen to be clauses, we recursively analyze these nested clauses. Thus linear structure of words within word groups, hierarchical structure of clauses and the functional structure of each of these clauses are clearly determined.

In all other grammar formalisms sentences are processed left to right. Human beings also process sentences linearly from one end to the other. How exactly human beings process sentences is not our immediate concern in UCSG. We do not make any cognitive claims. In UCSG we work from whole to part rather than one end to the other. If we process sentences from one end to the other, at any point in this process, decisions can only be taken based on the information gathered from the part that has already been seen. Working from whole to part has the advantage that we always have a global picture of what we are doing before we fill up the minor details. Working from whole to part implies exploiting the hierarchical structure. We determine the hierarchical structure first so that functional relationships can be understood separately at each level in the hierarchy.

The block diagram given below depicts the three main modules of UCSG and their inter-relationships. The following sections elaborate on these three modules of UCSG.



Sentence
L Grammar
Linear Structure analyzer
H Grammar
Initialized Chart
L–Structure
F Grammar
Hierarchical Structure Analyser
Functional Structure Analyser
H– Structure
F– Structure

2.3 Linear Structure

Linear position is significant in some situations and insignificant in other places. Grammars must ensure that the linear position constraints are enforced selectively. We make a distinction between linear position constraints within word groups and linear position constraints between word groups. The former type of constraints are universally found in all languages. It helps to identify word groups and permits the rest of the syntactic analysis to view the sentence as a sequence of groups. The latter is specific to positional languages. In UCSG, the F part of the grammar takes care of linear order of groups within a clause along with the constraints on other kinds of surface case markers. In the linear structure analysis phase, we are concerned only with the structure of word groups.

We have defined a word group as a structural unit that can as a whole play a role in some predication. Word groups occur both in positional languages and relatively free word order languages. They include elementary noun groups (that is, noun phrase not including relative clauses), prepositional groups, verb groups (sequences of auxiliary verbs and main verb) and adjectival and adverbial groups. In UCSG we equate prepositional groups with noun groups. The added preposition, a kind of surface case marker, only provides additional information for assigning a functional role to the embedded noun group. Some examples of groups in English are 'the computer' 'all the four kinds of content words', 'a man from the planet Mars', 'light green leaves', 'on the top of the mountain', 'could have finished', 'was about to fall', 'keeps crashing', 'very happy', and 'quite unimportant'.

It is very useful to identify these groups right in the beginning. All subsequent steps in syntactic analysis can then view sentences as sequences of groups rather than as sequences of individual words. The complexity of syntactic analysis is related to the length of the sentence. When viewed as a sequence of groups, the effective length of a sentence is reduced and hence all further syntactic analysis steps will become significantly simpler. The grammar becomes simpler and the parser more efficient.

It is very much possible that the same sequence of words group together in different ways in different sentences. In fact there may be more than one way of grouping words even for the same sentence. Some examples are given below.

41) I have broken pens

(*cf.* I have some broken pens with me.

cf. I have broken several pens earlier)

42) John is sleeping

43) Seeing is believing

44) *John is believing

45) Drinking water is available in plenty

46) Drinking water relieves thirst

47) I saw that gasoline can explode

In English a verb in the past participial form may form part of a verb group or it may function like an adjectival modifier in a noun group. Similarly an 'ing' verb may act like a gerund, a present participial or part of a present tense continuous verb. The ambiguity in 47) is because the word 'can' is ambiguous between a noun and a verb and the word 'that' is ambiguous between an adjective and a subordinating conjunction. The ambiguity in group formation in all these cases is due to lexical ambiguities. There can also be structural ambiguities. Consider

48) A notice was given by the PA to the Dean

(*cf.* A notice was given by the Registrar to the Dean

cf. A notice was given to the professor by the PA to the Dean)

In the above example 'by the PA', 'to the Dean', 'by the PA to the Dean' are all potential prepositional groups. This is a case of prepositional phrase attachment ambiguity. Whether it is lexical ambiguity or prepositional phrase attachment ambiguity, determining the correct way of grouping words requires the whole of syntax as well as semantic and pragmatic knowledge. We can gain the advantages of reducing the sentence from a sequence of words to a sequence of groups only if the process of identifying the groups is made very simple and local, without calling for constraints of hierarchical and functional structure. We would also like to avoid using large amounts of semantic or pragmatic information. We want our system to be essentially syntactic in nature.

We show that we can identify word groups using only local syntactic information. We will have to produce, however, all potential groups, not necessarily just the right ones. We will show that all potential word groups in a sentence can be obtained in a single linear scan of the sentence and in linear time. We will first discuss the internal structure of word groups in detail. We will then take up the process of linear structure analysis.

2.3.1 The Internal Structure of Word Groups

Consider the structure of noun groups such as 'the young boy'. Simple noun groups such as this can be specified as having an optional determiner, any number of adjectival modifiers and a head, *taken in that order*. The internal structure of such groups incorporates three aspects. Some items are optional. Optional items occur zero times or once. Some items can be repeatedly used. Repetition implies zero, one or more occurrences. Thirdly, the linear order of items is significant. Changing the word order renders the phrase ungrammatical or at least changes the meaning. These three aspects are completely sufficient to identify word groups. We use the term *linear structure* to denote the structure of word groups. Linear structure implies constraints on linear order, repetition and optionality. No other types of constraints are relevant.

Hierarchical structure, on the other hand, implies nesting of constituents one inside the other. Constituents may be nested in several ways and several levels deep. Hence the power of recursion is essential to deal with hierarchical structure. Recursion is more complex than repetition in general. Mere repetition of constituents does not require the power of recursion. Repetition can be captured using finite state machine power whereas recursive nesting requires the power of push down automata. We use the term hierarchical structure only if such recursively nested structures are involved.

There is no hierarchy in verb groups like 'has been coming' or 'is about to fall'. The entire group has a meaning but there is no hierarchy of meaningful units. Although the constituent words sometimes happen to be independently usable in English, 'has' in 'he has three pens' is completely different in meaning from the same word 'has' in 'he has been coming here regularly'. Similarly the word 'been' in 'has been coming' is completely different in meaning from 'been' in 'I had been to London'. We better simply treat these groups as atomic units.

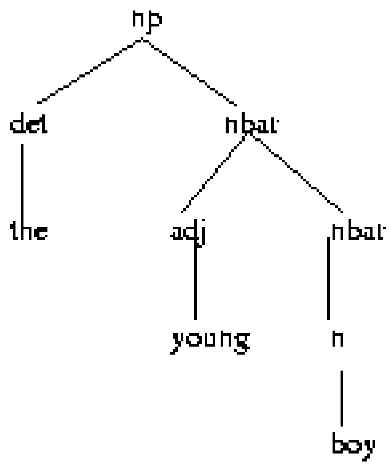
Is there a hierarchical structure in noun groups? Linear structure as well as hierarchical structure can be captured using phrase structure rules. However, phrase structure rules, whatever be the type, cannot depict unbounded repetition directly unless infinite in number, and repetition is always modelled using recursion. Thus both linear and hierarchical structure are often expressed using the same set of recursive phrase structure rules and one tends to miss the difference between the two. This has been the real source of confusion between linear structure and hierarchical structure in other grammar formalisms. The recursive nature of the phrase structure rules and the corresponding tree structures give us a *false* impression that we are capturing the hierarchical structure even when there is only linear structure and no real hierarchical structure. Rules such as

np → det nbar

np → nbar

nbar → adj nbar

nbar → n



and the corresponding tree structures for noun phrases like 'the young boy' fool us into believing that we are capturing the hierarchical structure within the np. No constituent is really nested one inside the other here, there is no recursion, there is no hierarchical structure. We just have an optional determiner, a sequence of adjectival modifiers and the head noun.

Of course there are more complex noun groups where there is really some hierarchical structure. In the word group 'the water meter cover adjustment screw' 'cover' is being modified by 'water meter' and not by 'meter' alone. 'adjustment' modifies 'screw' but 'cover' does not modify 'adjustment'. There is a nesting of modifier-modified relationships which can be shown using brackets : [[[water meter] cover] [adjustment screw]]. There is a hierarchy of constituents which can be seen easily in the nesting of brackets. For more examples consider 'English language syntax', 'on-line reservation system development team', 'computer communication systems design data handbook', 'data entry operator training department head', 'international educational academy', 'aluminium distilled water pumping system' and 'old ladies' hostel'.

How do we then capture the hierarchical structure within such groups? The answer is simple - we cannot. The hierarchical structure inherent within noun groups is a consequence of the modifier-modified relationships. Capturing the correct hierarchy of modifier-modified relationships requires a great deal of semantic and even pragmatic knowledge about the world we live in. Syntax can at most provide certain syntactic biases. Syntactic biases are preference guides at best [31, 42]. They can be easily overridden by semantic and pragmatic considerations. Capturing the modifier-modified relationships is beyond syntax and syntactic theories should recognize this fact. It is better not to pretend to do more than what is really possible. In general hierarchical structure within groups cannot be captured within a purely syntactic framework.

There is another source of confusion between linear and hierarchical structure. Noun groups may be modified by relative clauses. It has been traditional to consider the relative clause as part of the noun phrase. Thus noun phrases can have sentences within them and sentences of course have noun phrases within them. There is thus a recursive dependency between noun phrases and sentences. Other grammar formalisms have used recursive phrase structure rules for capturing this dependency. As a result, they lose the distinction between phrases and clauses.

Semantically the relative clause *is* a part of the noun phrase. However, in UCSG we make a distinction between the structure of word groups and the structure of clauses. Clauses are handled separately in the hierarchical structure analysis phase. Thus relative clauses are not relevant at all for linear structure analysis. Noun groups do not include relative clauses. To avoid possible confusion we have preferred the term noun group to the more common term noun phrase. During linear structure analysis we identify noun groups. During hierarchical structure analysis we recognize all the clauses and their interrelationships. Knowing that there is a relative clause adjacent to a noun group, we can obtain the proper interpretation of the two taken together during functional structure analysis. In UCSG there is no need for dealing with any kind of nested, recursive structure during the linear analysis phase. All aspects of recursive nesting are handled during hierarchical structure analysis. By definition, linear structure implies no recursion.

Other grammar formalisms, with the exception of Paninian Grammar, deal directly with words rather than with word groups. They attempt to capture linear as well as hierarchical structure within word groups at the same time, using a single set of grammar rules. Hence they use at least CFG power - more powerful than required for identifying the word groups. CFG power is anyway required to capture the hierarchical structure of clauses in a sentence and these grammar formalisms fail to see any difference between phrases and clauses. They mix up the two. Complete noun phrases may involve relative clauses and relative clauses in turn may have noun phrases within them. So the grammar becomes a lot more complex than really necessary. Phrases, which are in fact much simpler, are also handled using as complex a grammar as required to handle clauses. Secondly, instead of using sequences of groups, they use sequences of words, they process effectively longer sentences, and make hierarchical and functional analyses also a lot more complex. Finally, they pretend to be doing more than what they can really do, they miss out on the capabilities and limitations of syntax. They produce structural descriptions - tree structures in most cases, that can be deceptive.

Since groups are sequences of words, identifying groups requires knowing the words. We will therefore start with a sketch of some of the issues relating to words, lexicon and morphology.

2.3.2 Lexicon and Morphology

We assume at the outset, that there are identifiable units in the language called sentences and the sentences are made up of identifiable units called words. The input to our NLP system would be written text and thus the natural choice for a word is the orthographic word. In orthography, we write words separated by delimiters such as space, comma, period and exclamation mark. We define an alphabet set and then define a subset of this set as the delimiter set. A word is then any sequence of alphabets that does not include a delimiter and is separated by delimiters. We shall use this simple orthographic definition of a word in UCSG.

Words are often, although not always, the basic units of meaning. Words may also be composed of meaningful subparts ('teapot', 'noncommunicable', etc.). Morphology deals with the internal structure of words and how the meaning of the whole word is related to the meaning of its parts. While the words 'boy' and 'boys' are independent orthographic words, their meanings are systematically related to their internal structure. Thus we can provide a highly productive morphological rule for plural formation in English. Whenever meaning is compositional and productive morphological rules exist, only the basic forms should be stored directly in the lexicon and other forms are derived through appropriate morphological processes. When the meaning of the word is not simply a composition of the meaning of its parts or the rules are not productive, such word forms should be directly stored in the lexicon. Thus 'blueprint', 'return', 'good', 'better' and 'best' will all be directly stored in the lexicon. We introduce the concept of a *virtual lexicon* that includes the physical lexicon as well as the morphological component. Syntax always 'looks up' the virtual lexicon. Whether the word being looked up is directly stored in the virtual lexicon or is obtained through a morphological process is a matter purely internal to the virtual lexicon.

The Lexicon

In this section we will take up the question of what exactly do we need to say about the words in the lexicon. The lexicon has come to play an increasingly important role in syntactic theories. The demands on the structure and contents of a lexicon is to some extent theory bound. Each syntactic theory must specify the contents and structure of the lexicon that is part of the syntactic system.

We have taken the orthographic word as our basic unit. Hence the virtual lexicon will list all orthographic words in the language. The virtual lexicon will also associate each such word with grammatical information that will be useful for syntactic analysis. Since we are only dealing with syntax, we do not need meanings, usage, etymology, pronunciation and such other things.

Sentences in a language are practically infinite but we would like to describe the structure of these sentences using a finite grammar. We therefore need a mechanism for generalization. The first step in generalization is to categorize words into a small number of 'grammatical categories' and specify the grammar in terms of these categories rather than in terms of the words themselves. The lexicon stores the words, the grammatical categories, as well as other grammatical features like gender and number that may be useful in syntactic analysis.

How do we decide which set of grammatical categories to use? A computational model simply looks at categories as uninterpreted symbols. The lexicon specifies the categories for words and the grammar simply manipulates them as uninterpreted symbols. However, even the so called grammatical categories are inherently semantic in nature. We use language to express our thoughts, feelings and ideas about ourselves and about the world we live in. Whether we see the world through our language or we see the language through the world that we already know of, is not matter of concern for us here but it is important to realize the nexus between the words we use in our language and the things we perceive in the real world. This provides a semantic and hence universal basis for selection of basic grammatical categories.

There are certain objects, physical or mental, in the world and we use words to talk about them. Such words are traditionally called 'nouns'. There are actions and states in the real world that we can talk of and we use words categorized as 'verbs' to deal with them. We assume that the verb and noun categories exist in all languages. Additionally the languages may have 'adjectives' to describe things and 'adverbs' to describe actions and states. Thus categories like noun, verb, adjective and adverb seem to be universal. The universality of these kinds of grammatical categories follows from the nexus between the words and their real world interpretations, not from the role they play in specific syntactic theories. Apart from these basic categories, languages may show several other categories such as determiners, conjunctions and prepositions. These words have more of a grammatical function and are termed function words. Naturally, there are greater language to language variations in these categories. Function words play a very important role in determining the structure of sentences.

It is significant that linguistic theories like GB consider nouns, verbs, adjectives and *prepositions* as basic categories. These theories are heavily influenced by syntactic considerations. Preposition plays an important role in their theories of syntax and adverbs always get a very peripheral role in these theories. We believe that main function of a preposition is surface case marking - a kind of information that helps in assigning a functional role to the noun phrase that is attached to the preposition. Prepositions share this functionality with postpositions, morphological inflections and even linear position of constituents in a clause. Prepositions are not universal. Prepositions are more grammatical in nature.

Generalizations are essential to deal with a world that is large and complex, but over generalization has to be avoided. Throughout our developments in UCSG we will be concerned with this question of exactly how much to generalize for the task at hand. As far as recognizing the presence of a predication is concerned, knowing that 'sleep', 'eat' and 'give' are all verbs is sufficient. However these words differ in transitivity. Also, while 'eat' and 'know' are both transitive, 'know' takes a clausal object and ‘eat’ does not. These distinctions are useful for functional structure analysis. These are simply properties of these classes of verbs and we need to store this information in the form of ‘subcategorization frames’ in the lexicon. In UCSG, subcategorization frames specify what syntactic types of constituents must, may, or must not, appear in the different functional roles.

The subcategorization frames may include not only restrictions of syntactic types but also ‘selectional restrictions’ in the form of ‘semantic features’. Consider the sentences

49) ?I ate water

50) ?Water floats on oil

If we choose to say that 50) is grammatically valid, we don’t see any reason why 49) should not also be considered grammatical. In both cases, then we do not need any semantic features. If on the other hand, we say that 50) is also ungrammatical, we will need the knowledge of the entire world to be stored in our lexicon for segregating grammatical sentences from all the others. If the aim is to accept all and only valid sentences and always produce only the correct analyses, we may require unlimited amounts of world knowledge. Linguists have been using open ended sets of semantic features in their theories. Such theories are not computationally viable.

While the number of possible sentences in a language is extremely large and unbounded, we want a grammar that is finite and very much smaller to deal with the structure of all these possible sentences. Our emphasis here is not as much on segregating grammatically valid sentences from all possible random sequences of words, as on analyzing the structure of normal sentences. In UCSG we show that we can parse sentences and also limit the number of different parses produced to quite small numbers without resorting to large or unlimited amounts of semantic information and world knowledge. The key observation is that not all applicable constraints are equally useful in reducing the number of possibilities during parsing. A large number of constraints are very weak in the sense that they help only in very few situations.

How many bad sentences can density information help to avoid, compared to say, transitivity information? A few strong constraints applied in an appropriate manner makes UCSG computationally viable as well as efficient. The UCSG parser is essentially syntactic in nature. It uses only a few semantic features that greatly reduce the search and the number of parses produced.

The question of degree of generalization is relevant in the case of function words also. Note that the so called function words also contribute to the meaning of a sentence to some extent. While it is sufficient to note that 'in', 'of', 'with' are all prepositions for the purpose of recognizing prepositional phrases, this amounts to over generalization when the task is to assign a role to a prepositional phrase in the F-Structure. For example, while 'on', 'in' and 'at' can indicate space or time, 'with' and 'of' may not. The noun phrase

51) the appearance of the man under the tree with a broken branch near the edge of the road to the town with a market

is said to have 429 different parses [52]. This is clearly a consequence of improper use of generalization and specialization. In UCSG we understand that meaning distinctions in function words are crucial for the correct and efficient assignment of roles in the F-Structure. Hence it is not sufficient to specify in the lexicon that 'in', 'on', 'by' are all prepositions, we need to specify what kind of preposition each one is. For similar reasons, 'who', 'where' and 'when' are all words that may indicate relativization, indirect questions or direct questions but they are not all identical - they all have different meanings and this fact shows up when we look at the F-Structure of relevant kinds of clauses. Function words like subordinating conjunctions, relative markers and question words contain information regarding the presence of clauses, the type of clauses and even the clause boundaries. In UCSG these words therefore play a very important role. The lexicon should include all these relevant pieces of information about these words.

In summary, the lexicon in UCSG lists all the words and for each word gives information about grammatical categories, subcategorization, features relevant to grammatical agreement such as gender, number, person and case. Function words are shown with features useful for clause structure analysis and functional role assignment. In particular, prepositions, relative words and subordinating conjunctions are marked for their semantic roles. Content words may also show semantic features useful for functional role assignment.

Morphology

Morphology has several important roles to play. As we have discussed above, morphology is an integral part of the virtual lexicon. We would like to store only those word forms which cannot be easily derived from other words in the lexicon through some productive morphological processes. Morphology is used to recognize morphological variants of stored words. For example 'sing' would be stored and 'singing' would be derived by a highly productive rule in English. Similarly plural forms of nouns can be easily derived. Morphological analysis also provides useful information for forming word groups. Thus to recognize 'kept singing' as a single word group, we need to know that 'singing' is the 'ing' form of a verb. The most important role of morphology is, however, in providing information for assignment of functional roles to word groups. Morphological inflections are a type of surface case marking devices. The English requirement that the 'subject' appears before the verb is comparable to the Telugu requirement that the 'subject' be in the nominative case by default. Both of these requirements only help us to find out what is the 'subject'. Naturally therefore, morphology has a much greater significance in relatively free word order languages than in positional languages. For English only a few rules are essential from the role assignment point of view. Telugu morphology is very rich and a single word may enter into hundreds and hundreds of morphological formations [39]. We will not discuss the details of morphological analysis in this thesis. See [47, 45] for more details on Telugu morphology.

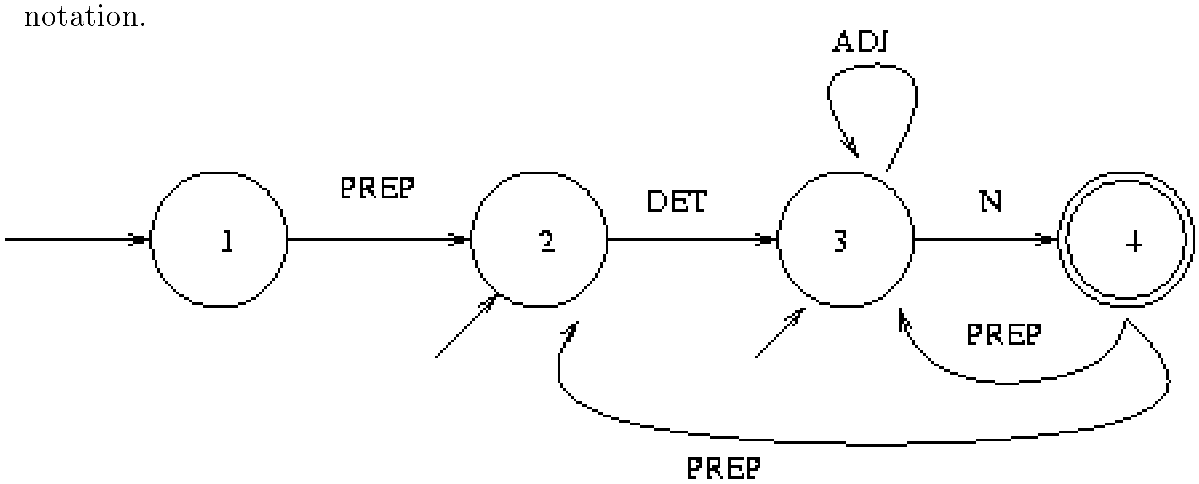
2.3.3 Linear Structure Analysis

In the linear structure analysis phase, we are concerned with the task of identifying word groups. We start with the given sentence, break it up into orthographic words using a predefined set of delimiters. We look up each word in the virtual lexicon. This involves checking with the physical lexicon and carrying out morphological analysis if required. As we keep looking up the words from left to right, we also build up the word groups. As we identify groups we note down grammatical features of the groups including surface case marking information which will be useful later for functional role assignment.

Linear structure implies constraints on linear order, repetition and optionality. No other types of constraints are relevant. It is important to realize that these three aspects of structure correspond to the three primary parts in the definition of regular expressions. Optionality is alternation with 'epsilon'. Linear order is concatenation. Repetition is Kleene closure. Because of the possibility of unbounded repetition, word groups can be potentially infinite. When infinite, this infiniteness comes purely from the unbounded repetition of certain substrings and not because of any recursive structure. Thus verb groups are finite sequences of auxiliary and main verbs. Noun groups in English include an optional determiner, an unbounded sequence of adjectival modifiers, a head and an unbounded number of 'pp' modifiers. Noun groups are thus potentially infinitely long.

Finite state machines can deal effectively with all these three aspects of structure namely optionality, repetition and linear order. Finite state machines can be deterministic or nondeterministic. We can either directly use the nondeterministic machine or convert it into an equivalent deterministic machine and use that for recognition. It is known that every nondeterministic finite state machine has a deterministic finite state machine equivalent to it. It is also known that recognition with finite state machines can be done in linear time. Linear time complexity implies that the entire process takes no more than a constant times the number of words in the given sentence. Dictionary look up and morphological analysis if any, are also done only once for each word in a given sentence. Hence the entire linear structure analyzer has a linear time complexity. Keeping in view that facts that linear order is always significant and linear time recognition is always possible, we use the term *linear structure* to denote the structure of such strings.

Writing the L grammar of a language is quite simple. Grammars can be written in any of the alternative representations such as state transition networks, state transition tables, regular expressions or regular grammar rules. All these forms are equivalent in terms of computational complexity. However, as we have seen above, regular grammars employ recursion to realize mere repetition and thus give a 'false' impression of hierarchical structure. We therefore do not recommend the use of regular grammar notation. We give below a sample L grammar for identifying prepositional groups in English. This grammar is written in the state transition diagram notation.



During linear structure analysis all potential groups in a given sentence are to be recognized. Groups may overlap one another. We also need to take care of lexical ambiguities. We give below an algorithm for identifying all potential word groups in a given sentence. This algorithm works for both deterministic and nondeterministic state transition diagrams. It simulates parallel processing. Instead of maintaining a single current state it maintains a current_state_set. Each word in the input sentence is considered only once and an amount of time bounded by the size of the grammar is spent per word. Hence the algorithm is linear in time complexity.

PSEUDO CODE FOR LINEAR STRUCTURE ANALYSIS:

```
MAIN()

    initial_state_set := [ ]
for each of the initial states s_i in the network do
      initial_state_set := union([(s_i,' ')],initial_state_set)
current_state_set := initial_state_set
step through the words in the given sentence and for each word w
advance(current_state_set,w)

      ADVANCE(current_state_set,w)

    new_state_set := [ ]
for each state (s,str) in current_state_set do
      for each out going arc a do
        if match(a,w) then {If any of the categories of w matches the arc}
          begin
            new_state_set := union([(end_state(a),concat(str,w))],new_state_set)
            if terminal_state(end_state(a)) then
            begin
```

output(concat(str,w))

new_state_set := union(new_state_set,initial_state_set)

end

end

current_state_set := new_state_set

Consider the phrase 'By the PA to the Dean' again. We give below a trace of our algorithm. The trace shows the sequence of current_state_sets.

```
[(1,''),(2,''),(3,'')]
By
[(2,'By')]
the
[(3,'By the')]
PA
[(4,'By the PA'),(1,''),(2,''),(3,'')]
Output : 'By the PA'
to
[(2,'By the PA to'),(3,'By the PA to'),(2,'to')]
the
[(3,'By the PA to the'),(3,'to the')]
Dean
[(4,'By the PA to the Dean'),(4,'to the Dean'),(1,''),(2,''),(3,'')]
Output : 'By the PA to the Dean', 'to the Dean'
```

Thus in a single linear scan of the sentence we can obtain all the potential word groups in linear time. Since in general no word in a sentence can be completely ignored, there cannot be any faster method. It is significant that no other grammar formalism has any component that is linear in time complexity, except possibly for the Local Word Grouper in the Paninian grammar formalism.

Thus the L module separates out a part of grammar that is relatively simple and amenable to extremely efficient parsing. By separating the L part we have not only achieved simpler and more efficient processing of the L part itself but also significantly simplified the subsequent levels. The subsequent processes only need to look at sequences of groups rather than sequences of words. Although the entire grammar of any human language may not be within the scope of regular grammars, languages do have significant parts that can be captured using regular grammars. Proof or belief that grammars of a particular complexity are insufficient to deal with the whole of the syntax of human languages has always been taken earlier to imply the need for a more complex grammar for the entire language. We think it is wiser to divide the problem so that parts can be handled in the simpler grammars and more complex grammars are used only where really essential.

Different languages use completely different sets of words and quite different sets of morphological rules since the relationship between word form and meaning is completely arbitrary. This fact also favours an independent L component. By separating out language specific parts, we get to see the real universal core grammar.

2.4 Hierarchical Structure

We will start this section with a detailed study of clause structure. We will then see how exactly we capture the hierarchical structure of clauses in UCSG. We will provide the complete set of H grammar rules for English and then show that parametric variations of these same rules hold good for Telugu and other languages. We will also discuss the issue of computational complexity of hierarchical structure analysis.

2.4.1 Types of Clauses and Clause Structure

A sentence may consist of one or more predications. To know the structure of a sentence we have to know the internal structure of each of these predications as well as the relationships between these predications. Each predication must have a verb group along with the various other participants to fully specify the action, state or event being described in that predication. Some of the participants in a predication are optional while others are essential to give complete sense.

In UCSG we view clauses as a surface representations of predications. There is a one-to-one correspondence between predications and clauses. Every clause includes exactly one verb group, all the essential arguments as also all the optional modifiers, thereby conveying the full meaning of the predication. Clauses are therefore appropriate units for structural description. Identifying clauses and determining the hierarchical structure of clauses in a sentence is a very important aspect of syntax.

The participants in a predication indicate the doers, the done-tos, the place, time, etc. By default, we use nouns to denote persons, things, places, times, etc. Naturally, the participants in a predication manifest themselves as noun groups by default. It may be recalled that prepositional groups are equated with noun groups in UCSG. The added preposition is a surface case marker - it only provides extra information for assigning a functional role to the noun group. Unless stated otherwise, our use of the term noun group should be read as including prepositional groups.

However, an entire predication can also be viewed as a 'thing' and hence we often use clauses in place of noun groups. This leads to a hierarchical nesting of clauses. It is possible to 'talk about' an entire predication (Example 52) - the subject of a clause can be a clause. There are a number of verbs relating to mental actions such as knowing, observing, wanting and believing where the "thing" one knows, observes, wants or believes can be a predication in itself (Examples 53 and 54). So clauses can be objects of other clauses. The place where an action or event takes place can be described in terms of another action or event that is related to that place. So is time. The reason or purpose behind a predication is often specified as another predication. Thus modifier roles can also be clauses (Examples 55 and 56). In all these cases one clause acts as a participant in another clause - one clause fills a role in the F-Structure of another clause.

52) That the company kept the secret for so long annoyed everyone

53) We all know who makes these chips

54) We did not expect that this company would do such a thing

55) After the bug in the new chip was announced, the company's sales have dropped sharply

56) The company's sales have dropped sharply because it cheated its customers.

The noun groups that take on participant roles and fill the slots in the F-Structure can be further modified by clauses. These clauses are called relative clauses (Example 57). Relative clauses do not fill the main slots of an F-Structure. They do not take any participant role by themselves in any predication. They only add more information about a noun group that fills one such role. This distinction between clauses that fill particular functional roles and clauses that modify the fillers of roles in a predication is very important in UCSG.

57) Computers which include the new chip are not reliable

The possibility of using clauses as participants or modifiers of participants within another clause implies that clauses in a sentence may be nested one inside the other recursively. This leads to a tree of clauses. A simple sentence makes a single predication, it has only one verb group and only one clause. In this case the clause structure tree would be just the root node. In complex sentences the clause structure tree can be several levels deep. Each clause except the root clause has a parent - the clause in which it either takes on a participant role or modifies one such participant role. Thus clauses are always related to one another in a tree fashion. Hierarchical structure analysis therefore requires the power of recursive rules. Repetition is not sufficient. This fact distinguishes hierarchical structure from linear structure.

Nominalized Clauses

We have stated above that clauses can substitute for noun groups as fillers of roles in an F-Structure. In fact, clauses have a general tendency to become nominalized.

Consider the following examples.

58) After I saw the new machine I fell for it.

59) After seeing the new machine I fell for it.

60) On seeing the new machine I fell for it.

61) To see the new machine is to fall for it.

62) At the sight of the new machine I fell for it.

In 58) 'I saw the new machine' is a finite clause - it can stand on its own as an independent sentence. In 59), 'seeing the new machine' has become nonfinite. It is a little more like a noun group. Correspondingly, the subordinating conjunction 'after' has become more like a preposition. See 60) - 'after' is similar to the preposition 'on'. 61) shows another kind of nominalized clause in English. In 62) the process of nominalization has reached the limit - the verb 'see' has changed completely into the noun 'sight'. The object of the transitive verb has become a prepositional group modifier. The preposition 'at' is making the whole noun group into a prepositional group. Thus 'sight' has acquired a distinct noun character while retaining some of its original verb nature. *Nominalized clauses exhibit dual nature - they are clauses and noun groups at the same time.*

The acquired noun character permits 'love at first sight', 'the sight of the machine' as well as 'My sight of the machine' while 'I see' and 'see her' are incomplete. Thus nominalization permits otherwise mandatory participants to be missing. Completeness criteria cannot be applied directly to nominalized clauses. The explanation is simple if one takes a pragmatic view. The sentences 58) to 62) above do not all mean the same thing and in practice we need to be able to say any of these. We use nominalized clauses only when we prefer not to emphasize some of the participants, only because we prefer not to mention them explicitly, only where we wish to emphasize the nominal nature of these clause rather than their clausal nature. What is important is the nominalized nature of these clauses, not whether the clause is finite or nonfinite. Nonfiniteness follows from the fact that some of the participants are de-emphasized, made optional or even completely omitted. The fact that nonfinite verbs do not have to agree with the subjects in English is simply a consequence of the fact that there may be no subjects to agree with.

The types of nominalized clauses vary from language to language but this nature of clauses to become nominalized seems to be universal. Nominalized clauses are clausal substitutes for noun groups. Naturally, they are clauses and noun groups at the same time. Hence nominalization of clauses has important repercussions to any syntactic system. When should we look at a nominalized clause as a clause and when should we look at it a simply a noun group? In UCSG we exploit this dual nature of nominalized clauses to achieve our aims of universality, simplicity and elegance of the grammar and efficiency of parsing. We will see that there are several significant advantages for treating nominalized clauses as noun groups while dealing with the hierarchical structure of clauses.

2.4.2 Clause Structure and F Constraints

In UCSG, recognizing clauses and their hierarchical structure - the H level, precedes the F level wherein the functional roles played by the various constituents are analyzed. The aim of hierarchical analyzer is to recognize all the clauses in a given sentence and to determine their nested relationships. We also wish to determine the clause boundaries so that we can localize the functional structure analysis to the respective clauses. In a sentence with n clauses, this gives the effect of analyzing n simple sentences rather than one complex sentence with n clauses in it.

Determining clause boundaries requires finding out which groups are part of which clauses. Whether a given group can be a part of a given clause depends on the functional structure constraints such as subcategorization frame, selectional restrictions, linear position and other surface case markers and agreement restrictions. Since there can be several clauses in a sentence and a number of potential groups, hierarchical structure analysis seems to be a complex task requiring almost all of the constraints of the F level to be applied. There will be no real advantage of localizing functional structure analysis if hierarchical structure analysis itself requires the application of all of the functional structure constraints over the entire sentence. We need a simpler technique. We should be able to analyze the hierarchical structure of clauses without the need of functional structure constraints.

We have noted that the participants in a predication are noun groups by default. Functional structure constraints relate to the participants in a clause. So, not considering functional dependencies between the participants of a clause means not considering the noun groups. The functional dependencies among the participant groups in a clause do affect the clause boundaries. But these constraints primarily deal with completeness and internal consistency of a clause and clause boundary determination is only an indirect consequence of these completeness and consistency constraints. It is not appropriate to apply functional constraints for clause boundary determination. That would be like asking for the solution to a whole problem in order to solve a part of it.

There are several important benefits for not considering noun groups and functional dependencies among groups. Ignoring the noun groups makes the effective length of the sentence much smaller. Noun groups need not be reflected in the hierarchical structure part of the grammar, which makes the grammar much simpler than it would be otherwise. Both of these factors taken together lead to a very efficient parsing for the H phase. Since nominalized clauses also have a noun group character, one has the option of treating them as noun groups and thus ignoring them completely from hierarchical structure analysis. Treating nominalized clauses as noun groups and ignoring them for hierarchical structure analysis has many advantages as we will see soon. This option finds use in dealing with cross serial dependencies in Dutch, for example. Ignoring noun groups at the H level also has major implications for parsing relatively free word order languages.

Functional structure constraints are weaker and indirect constraints on clause boundaries but they are essential for determining the clause boundaries completely. If we ignore the noun groups and all the F constraints, the clause structure analysis problem will no longer be fully solvable. The clause boundaries can be determined only partially and hence functional structure analysis also cannot be completely localized to one clause at a time.

However, a sentence is *nearly-decomposable* into the various clauses in it. We will show that while the clause boundaries can only be determined partially, we can analyze the hierarchical nesting of clauses fully without looking at any noun groups and without applying any of the functional structure constraints. The interactions between clauses are limited and local. The F-Structure analyzer can still be localized to one clause plus possibly a limited region within one adjacent clause. Groups within other clauses do not even have to be considered. Hence H-Structure analysis significantly simplifies F-Structure analysis. We will show that H-Structure analysis itself can be done very efficiently and so the parser as a whole becomes efficient. Let us now see how exactly we can analyze the clause structure without considering noun groups, nominalized clauses and functional structure constraints.

2.4.3 Verb Groups, Sentinels and Clause Structure

It is not wise to apply functional structure constraints to determine the clause structure. That would mean too much of effort for what is really gained. Fortunately, human languages provide much more direct clues for clause structure analysis. Human languages tend to explicitly mark one of the ends of each clause in a sentence with certain markers. Subordinating conjunctions such as 'that', 'after' and 'because' and relative pronouns such as 'who' and 'which' are examples of such markers. We call such markers as *sentinels*. In UCSG we believe that the most important thing about such words is their role in signalling the presence of clauses, indicating the type of clauses and marking the clause boundaries.

In general, every clause in every sentence would be well marked for one of its two boundaries. We should add immediately that exceptions to this rule exist. There are basically two kinds of exceptions. Firstly, there are some types of clauses where there are no overt sentinels. English gerundial and infinitival clauses are examples. Secondly, there are sentences where the sentinels marking the clause boundaries are optionally dropped. We will deal with the rule first and defer the treatment of exceptional cases to a later section.

If we ignore participant groups such as noun groups, adjective groups and adverb groups, a sentence will be reduced to a sequence of verb groups and sentinels. Verb group-sentinel sequences have some nice properties. By definition, every clause has a verb group in it. Also, as a rule one end of every clause is marked by a sentinel. There is thus a balance of verb groups and sentinels. Verb groups and sentinels behave very much like left and right brackets. The brackets must match - clauses in a sentence always follow this proper nesting constraint. By simply counting and matching the verb groups and sentinels from left to right we can know a great deal about the clause structure. For example we can very easily determine the main clause verb group. This makes it possible to start the functional structure analysis from the main clause and recursively analyze the nested clauses as we go along.

Consider the following English examples. In English relative clauses, clausal subjects, clausal complements and modifying subordinating clauses are all marked for their beginning by sentinels. The main clause has no explicit sentinel. We write down the verb group-sentinel sequence separately for convenience. We start with a count of one and scan this sequence from left to right. We increment the count every time we get a sentinel and decrement the count every time we get a verb group. The first time we get a count of 0, we are at the main clause verb group.

63) The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night

believed	that	who	wanted	must-have-escaped
0	1	2	1	0

64) That the new computer which the company that everyone trusted had sold to me was defective has been annoying me because I had decided to use this computer for all my investigations in future.

That	which	that	trusted	had-sold	was-defective	has-been-annoying
2	3	4	3	2	1	0

because had-decided

1 0

In the first example above, the main clause verb group is 'believed' and in the second example, it is 'has-been-annoying'. Thus determining the main clause verb group is extremely simple. For any grammatical sentence, the count should never become negative and the final count must be 0, indicating proper matching. Thus the verb group-sentinel sequence carries important information about the grammaticality of the sentence too. Also observe that in both the examples the 'sentence' we consider for hierarchical structure analysis - the verb group-sentinel sequence, is only about one fourth of the original sentence in size. We will see Telugu examples in a later section.

2.4.4 Determining Hierarchical Structure

To determine the hierarchical structure, we make use of the of the groups already obtained in the linear analysis phase. We have seen the correspondence between a clause and a predication and we have noted that each clause must therefore have exactly one verb group within it. Since all the verb groups will have already been obtained in the linear analysis phase we will know how many clauses are there in the whole sentence. We also know a hinge point within each of these clauses - the verb group itself. What remains to be done, therefore, is to determine the relationships between clauses and the clause boundaries.

It is the main thesis of UCSG that the hierarchical structure of clauses within a sentence is primarily determined by these sequences of verb groups and sentinels and not by the functional structure constraints of completeness and consistency. There are very strong constraints on these verb group-sentinel sequences. Applying these strong constraints leads to simpler grammars and much more efficient parsers than would be possible if the indirect and weaker functional structure constraints were employed. The constraints on verb group-sentinel sequences are sufficient for identifying all the clauses and their linear and hierarchical relationships. Clause boundaries are also determined, although only partially.

There are parametric variations across languages as to which types of clause are marked for their start and end positions. Thus relative clauses are well marked for their start in English while they are well marked for their end position in Telugu. In English, clausal subjects, clausal objects, modifier clauses as well as relative clauses are all overtly marked by sentinels for their beginnings. The main clause in English ends either at the end of the sentence or at the start of a post verbal modifier clause (whose beginning will be marked explicitly by a sentinel). Note that while such a modifier clause has a role to play as a whole in the matrix clause, groups and clauses within the modifier clause do not form part of the matrix clause directly. In general sentence boundaries also act as clause boundaries and are therefore similar to sentinels in their function. Since sentence boundaries are normally clear cut, languages tend to use no explicit sentinels in these positions. In summary we can say that one end of all these types of clauses is always very easily and completely determined.

Sentinels are not completely sufficient for correct determination of all the boundaries of all the clauses in a sentence. However, these clues are sufficient to determine one end of each clause precisely while the other end can also be bounded within two limiting positions. Let us call the area between these two limiting positions as *grey area*. Functional structure analysis can then be localized to one clause plus one grey area, if any, between that clause and a neighbouring clause. We will call this whole area as the *local domain* for that clause. The local domain for a clause starts from the exactly known boundary, passes through the verb group and ends at the far end of the grey area on the other side of the clause. Constituents falling outside this local domain need not even be considered. The localization effect is thus quite significant.

In summary, we know the number of clauses in a sentence, the verb group of each of these clauses and one of the clause boundaries as indicated by the sentinel of that clause. We therefore need to determine the other clause boundary and the hierarchical structure of clauses. In the next section we will develop the H part of the grammar of English and show how the clause boundaries can be determined. We will then show how the same grammar rules with parametric variations will handle the clause structure of Telugu, a relatively free word order language. Then we will discuss computational complexity of hierarchical analysis. We will finally take up exceptional constructions where sentinels are either optionally or inherently absent.

The Hierarchical Structure of English Sentences

We make distinctions between three types of clauses in a sentence in English. Let us name them f_clause, sub_clause and rel_clause. Note that we have not included the nominalized clauses. As observed earlier, we can treat these nominalized clauses as noun groups and ignore them at the hierarchical analysis level. Every sentence must have one and only one f_clause excluding the f_clauses if any in sentences embedded within it. Other two types of clauses are purely optional. The f_clause is without any explicit sentinel. Relative clauses, named rel_clause, are modifiers of participants in a clause and they start with the sentinel denoted rl. Sub_clauses fill the participant roles. Thus clausal subjects, clausal objects and the clausal modifiers of place, time etc. are all termed sub_clauses. Sub_clauses begin with the sentinel named sb. Thus every clause has one verb group, denoted vg, and every clause except the f_clause is marked by a starting sentinel, either sb or rl. At H-Structure level our input 'sentences' would include only vg, sb and rl and nothing else.

We have chosen the term 'sub_clause' to be indicative of subordinate clause but our definition includes modifier clauses as well as clausal subjects and complements. The distinction between modifier clauses and clausal subjects and complements cannot always be made based only on internal requirements. Thus 'when the bell rang' can be a time modifier, a clausal subject as well as a clausal complement. All that we can do, and we need to do, is to identify this as a sub_clause. The external requirements will be applied at the F-Structure level to determine the role played by this clause in the enclosing clause.

Since we do not consider noun groups at all in the H part of the grammar, we do not need to make distinctions between clauses with and without missing participants. The fact that relative clauses and indirect questions have missing participants while 'that clauses' do not have any, is therefore irrelevant to us here. Indirect questions are also simply sub_clauses for us.

The distinction between rel_clauses and sub_clauses is important. While sub_clauses take up participant roles, rel_clauses are only modifiers of other participant roles. Thus rel_clauses can be *taken apart*, bringing the participants of a single clause close together. There will no longer be any *long distance* dependencies arising out of the insertion of relative clauses.

With these things in mind we can now write down the H grammar rules for English. A sentence s could be either the f_clause alone or f_clause with sub_clauses which in English may appear before the f_clause, after the f_clause, or in both positions. There can also be more than one sub_clause in a row. For example, there can be two sub_clauses before the f_clause verb group and either both are modifier clauses or one is a clausal subject and the other is a modifier clause. Similarly there can be clausal object and/or modifier clauses after the f_clause verb group. Hence the rule for s can be written as

Rule 1 s → (sub_clause)* f_clause (sub_clause)*

where the '*' indicates zero or more occurrences. The f_clause itself would either be a vg alone or a vg with rel_clauses before the vg (pre-verbal noun group being modified by a relative clause), after the vg (post-verbal noun groups being modified by relative clauses), or in both positions. Thus the rule for f_clause will be

Rule 2 f_clause → (rel_clause)* vg (rel_clause)*

The rule for rel_clause is simple. It has just the sentinel rl followed by a sentence s. The rule for sub_clause is similar except that the sentinel would be sb.

Rule 3 rel_clause → rl s

Rule 4 sub_clause → sb s

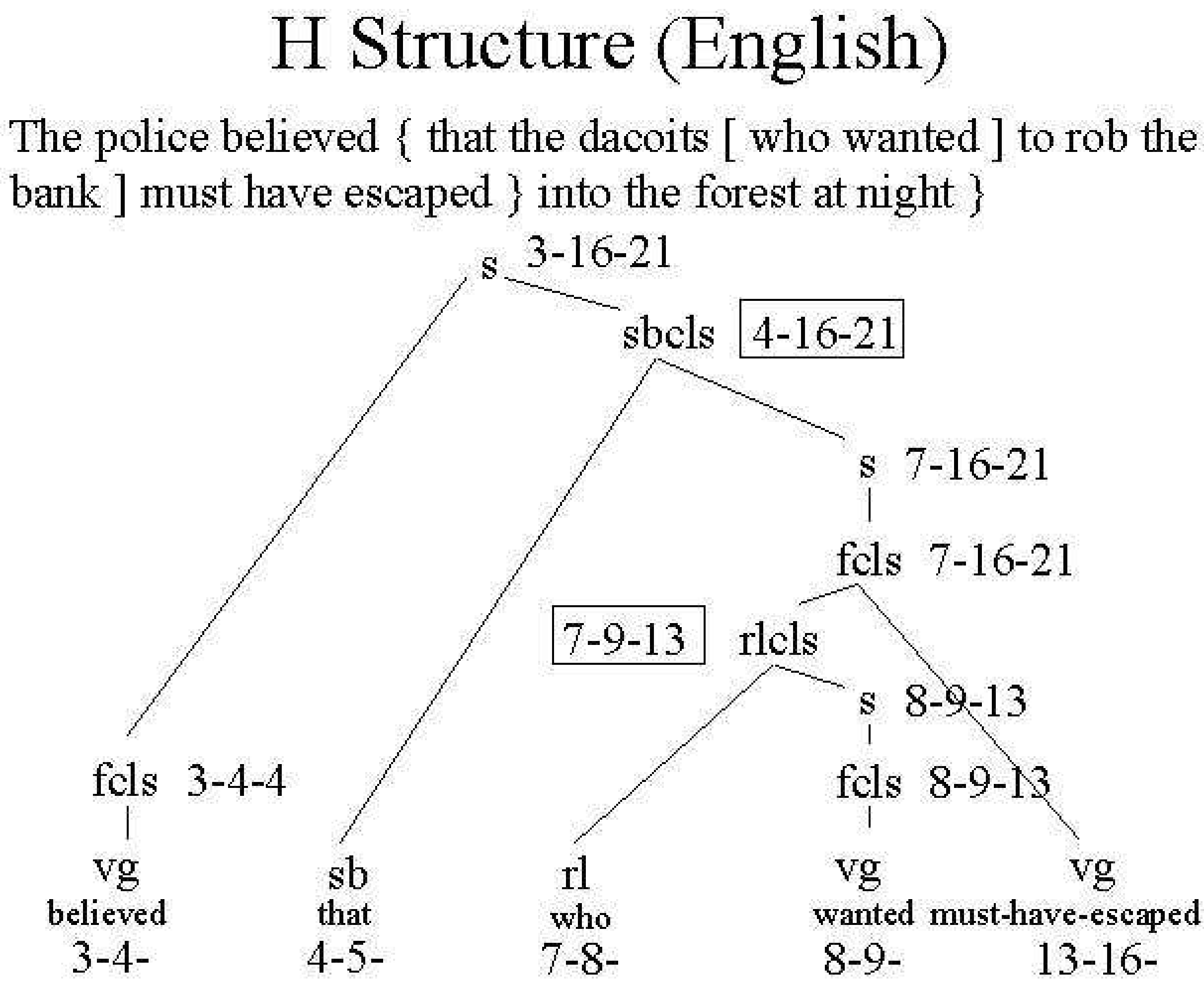
Note that the '*' operator in rules 1 and 2 is used to handle two or more independent, non nested sub_clauses or rel_clauses in a row. All kinds of recursively nested clause structures are obtained by the appropriate expansions for the s in the rules 3 and 4. Let us see how exactly these rules work. Consider again the sentence

65) The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night.

Linear structure analysis gives us all potential groups. We are now interested only in verb groups and sentinels. The input to the hierarchical structure analyzer is therefore the string 'vg sb rl vg vg' which corresponds to the sequence 'believed that who wanted must-have-escaped'. The four rules above give us the following tree structure:

```
[s,[f_clause,[vg]],[sub_clause,
   [sb],[s,[f_clause,[rel_clause,
      [rl],[s,[f_clause,[vg]]]],
         [vg]]]]]
```

We will now specify how the clause boundaries can also be determined. Note that clause structure trees are ordered trees. The linear order of child nodes is significant. This is true of both positional languages and relatively free word order languages. Hierarchical structure implies ordered tree structure - both recursive nesting and linear order of child nodes are significant. In an ordered tree, the spans of a constituent are determined by the spans of its child nodes. Note that clause structure trees do not include all the words in the input sentence. Therefore, we can only assert that every constituent must span at least from the starting position of its left most child and go at least till the position where its right most child ends. At the leaf nodes, we will have terminals like vg, sb and rl whose spans are known from linear structure analysis. Hence by propagating this minimal span information up the tree we can obtain the minimal spans of all the constituents in the tree during parsing itself. The result for our example is shown below. The i^{th} word in the sentence is taken to go from position i to position (i + 1).



```
[(s,3,16),[(f_clause,3,4),[(vg,3,4)]],[(sub_clause,4,16),
   [(sb,4,5)],[(s,7,16),[(f_clause,7,16),[(rel_clause,7,9),
      [(rl,7,8)],[(s,8,9),[(f_clause,8,9),[(vg,8,9)]]]],
         [(vg,13,16)]]]]]
```

Since our s has an f_clause and a sub_clause and since the f_clause begins at position 3 and the sub_clause ends at 16, the s must at least span from 3 to 16. The sub_clause in turn must go up to 16 because the s in it goes up to 16 and so on. Since in English the sentinels mark the beginnings of clauses, we have fully determined the starting positions of all sub_clauses and rel_clauses. We have also obtained the earliest points where they can end. These points mark the left end of the respective grey areas.

Determining the right end of the grey areas is also very simple. The matrix s spans right up to the end of the sentence. Child nodes of a given node may span all the way up to the end of the parent node or up to the start position of a right sibling if any, whichever is earlier. Thus by propagating the right most limiting positions down the hierarchy, we can obtain the right ends of the grey areas. The result is shown below. Observe that for terminals the spans are exactly known and there are no grey areas.

```
[(s,1,16,21),[(f_clause,3,4,4),[(vg,3,4,_)]],[(sub_clause,4,16,21),
 [(sb,4,5,_)],[(s,7,16,21),[(f_clause,7,16,21),[(rel_clause,7,9,13),
  [(rl,7,8,_)],[(s,8,9,13),[(f_clause,8,9,13),[(vg,8,9,_)]]]],
   [(vg,13,16,_)]]]]]
```

For convenience let us denote sub_clauses by curly brackets and rel_clauses by square brackets. We attach numbers to brackets to help match the corresponding left and right brackets in sentences which may have several rel_clauses and sub_clauses.

Knowing the boundaries of rel_clauses and sub_clauses, we can easily see the boundaries of s and f_clauses too. We do not use any explicit brackets for s and f_clauses for the sake of clarity of representation. The hierarchical structure of the above sentence, depicting the recursive nesting of clauses as well as the clause boundaries can then be shown as follows

The police believed {0 that the dacoits [0 who wanted 0] to rob the bank 0] must have escaped 0} into the forest at night 0}

Thus the local domain of the sub_clause extends from 'that' till the end of the sentence and the words 'into the forest at night' are in the grey area of this clause. The groups in the grey area may or may not be a part of the sub_clause. Only functional structure analysis will tell this. Similarly, the local domain of the rel_clause is from 'who' to 'bank' inclusive and the grey area includes the words 'to rob the bank'. The local domains of sub_clauses and rel_clauses start from the sentinel at the left and terminate at the far end of the grey area on the right side of the clause. On the other hand, f_clauses terminate at the beginning of the following sub_clauses if any, or at the end of s, both of which are fully determined. If there is any sub_clause on the left, an f_clause may have a grey area to the left, which coincides with the grey area of the sub_clause.

The four rules given above are sufficient to deal with the hierarchical structure of clauses in English. We can determine the hierarchical structure of clauses as well as the clause boundaries. For every clause the local domain and the grey area if any, can be determined. We thus have a very small and an extremely simple set of rules. Unlike in many other formalisms, we can claim that these rules form *a complete set* in UCSG. These are not mere samples, these are the only rules that we will need for English.

Note that all our rules except for the rule for s are lexicalized - they hinge upon a lexical item such as rl, sb or vg. As far as the main clause is concerned, we have already seen that the verb group around which the main clause hinges is also easily determined by simply counting and matching vgs with sbs and rls. Rules 1 and 2 use only iteration and not recursion. The recursion in the rules 3 and 4 account for different possible nested structures. These rules are ambiguous. A given sequence of vgs and sentinels can have more than one possible H-Structure and our rules can generate all possible H-Structures. Consider the sentences

66) Increased foreign investment in the country is being attributed to the policy changes which the central government has made after P.V. Narasimha Rao took over as the Prime Minister.

67) Increased foreign investment in the country is being attributed to the policy changes which the central government has made after India Today published an article on the effects of the new economic liberalization.

In 66) the sub_clause would preferably be attached to the relative clause whereas in 67) it would rather get attached to the matrix clause. Since different possible attachments exist, the grammar will have to be ambiguous - it must be able to produce more than one structure for a given sentence.

Note however, that the ambiguity in the grammar is limited and localized to the recursive call of s in the rules 3 and 4. These recursive calls are essential to deal with the different possible nested structures of clauses. The main source of ambiguity here is sub_clause attachment, a predominantly semantic problem. The best that an independent syntactic module can do is to produce all potential structural descriptions. Of course syntax can provide syntactic biases in some cases but such purely syntactic biases are weak assertions in the sense that they may be very easily overridden by semantic preferences. In this thesis we will not address issues of syntactic bias. We will be content with a grammar that can produce all the possible attachments.

2.4.5 Universality and Parametric Variations

The grammar rules we have developed for English, or parametric variations of these same rules, will hold good for other languages also. We will now show this for Telugu, a language of southern India characterized by relatively free word order. This will also give us an idea about the nature of parametric variations that we can expect in general. After that we add a note on hierarchical structure analysis of Dutch, a language which exhibits cross serial nestings.

Consider the following Telugu sentence which is equivalent to the English example we used in the previous section:

68)

baaMkunu doocaalanukonna doMgalu raatri aDavilooniki paaripooyi uMTaarani pooliisulu nammaaru

doocaalanuko +	ina	paaripooyi uMTaaru +		ani	nammaaru
vg	rl	vg		sb	vg
0	1	0		1	0

With the same counting technique that we used for English, we can identify the main clause verb group. The only difference is that in Telugu the right most position where the count becomes zero indicates the main clause verb group. The count can never become negative.

Let us now formulate the H grammar rules for Telugu. Telugu is a verb final language. All the sub_clauses and rel_clauses appear before the vg. We disallow sub_clauses appearing after the f_clause thereby simplifying rule 1 to

Rule 1 s $\rightarrow$ (sub_clause)* f_clause

Similarly we disallow rel_clauses after the vg simplifying rule 2 to

Rule 2 f_clause $\rightarrow$ (rel_clause)* vg

In Telugu the sentinels rl and sb mark the end points of the respective classes instead of the starting points. Hence rules 3 and 4 take their parametric variations

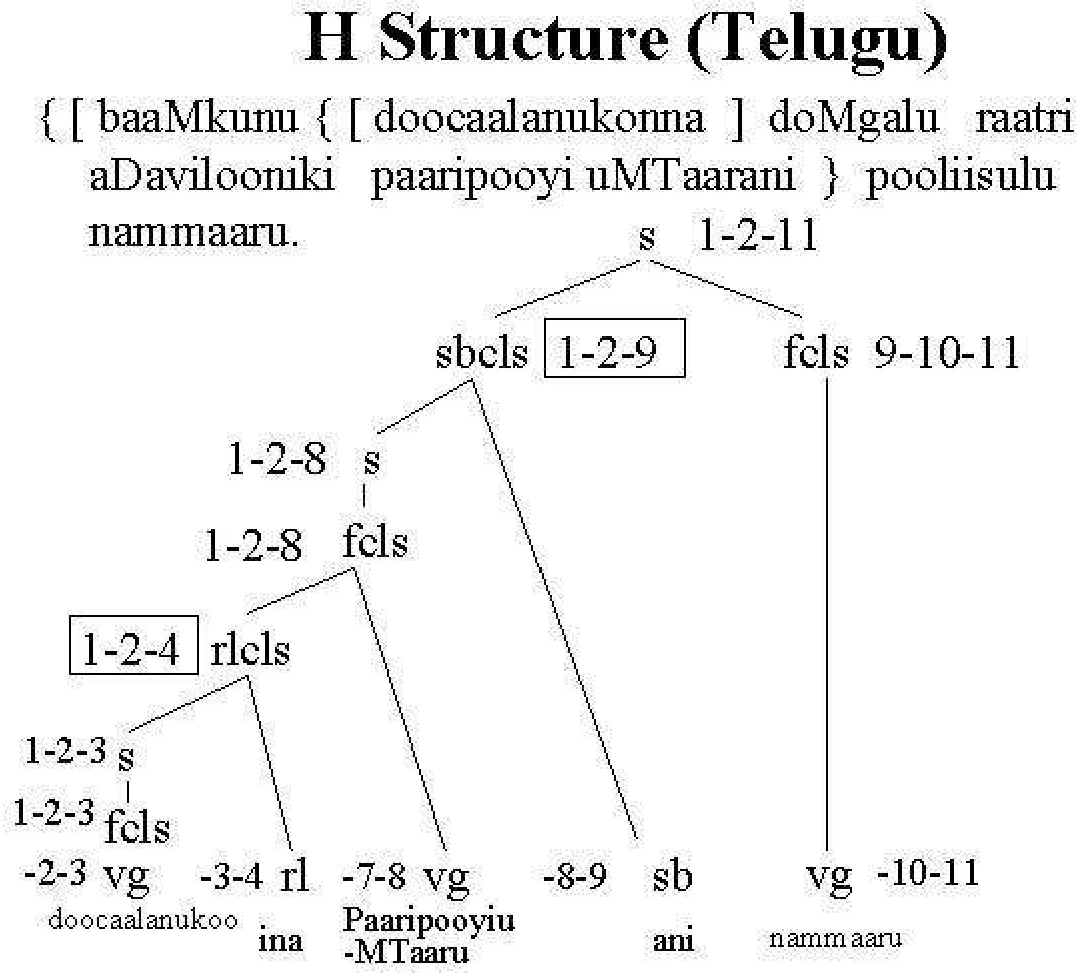
Rule 3 rel_clause $\rightarrow$ s rl

Rule 4 sub_clause $\rightarrow$ s sb

We need no more changes. These parametric variations in the four rules suffice to deal with the hierarchical structure of Telugu. We still have only four rules. The rules are similar and even simpler than the English rules. As far as H-Structure is concerned, there is no difference between positional languages and relatively free word order languages. The concept of relatively free word order only applies to the noun groups none of which are relevant to H-Structure analysis. Telugu verb groups and sentinels indicate clause structure the same way as in English, they follow the same kind of grammar rules. The clause structure of the sample Telugu sentence is given below.

We have seen in chapter one that Dutch exhibits cross-serial nesting of clauses. Clause structure analysis is based on the proper nesting principle. Cross-serial nesting implies improper nesting of clauses. How do we handle Dutch then?

We have noted in chapter one that only infinitival clauses - a type of nominalized clauses, can enter into cross-serial dependencies in Dutch. We have also argued for excluding nominalized clause for H-Structure analysis giving several advantages of doing so. Thus Dutch requires no changes whatever. We simply postpone the analysis of nominalized clauses to functional structure analysis phase just as we did for English and Telugu. The grammar of hierarchical structure seems to be universal, language invariant.



2.4.6 Computational Complexity and Efficiency of Parsing

The main aim in searching for better grammar formalisms is to discover the simplest grammars that are sufficient to deal with the syntax of natural language sentences. The simplest grammar would not only be theoretically the most satisfying but also the least powerful and hence computationally the most efficient. We will first show that CFG power is sufficient for hierarchical structure analysis. We will then go on to argue, based on certain justifiable assumptions, that hierarchical structure analysis can also be reduced to linear time complexity.

Note that all the four rules we have given above have single nonterminals on the left hand side and nonempty sequences of terminals and nonterminals on the right hand side. Thus all our grammar rules are of the context free power. The rules are not exactly in the CFG format however, because of the use of the star operator which signals repetition. We have seen that CFG rules capture iteration also through recursion thereby imposing more structure than there is. If we ignore this issue of extra structure, we can rewrite the above rules in the form of CFG rules. If we wish to avoid using epsilon rules (rules with empty right hand sides), we can rewrite our rules in the form of the following 12 CFG rules :

Rule 1a s → f_clause
Rule 1b s → sub_clause f_clause
Rule 1c s → f_clause sub_clause
Rule 1d s → sub_clause f_clause sub_clause

Rule 2a f_clause → vg
Rule 2b f_clause → rel_clause vg
Rule 2c f_clause → vg rel_clause
Rule 2d f_clause → rel_clause vg rel_clause

Rule 3a rel_clause → rl s
Rule 3b rel_clause → rl s rel_clause

Rule 4a sub_clause → sb s
Rule 4b sub_clause → sb s sub_clause

It must be emphasized that these CFG rules are only weakly equivalent to our original grammar rules. They accept or reject the same set of sentences as our original grammar rules do, but the recursive rules 3b and 4b can produce incorrect and unintended nested structures. In the CFG format we have more rules, three times more. Also, generalizations are lost. Expanding the nonrecursive rules 1 and 2 into four rules each amounts to loss of a simple generalization that some items occur zero, one, or more times. Nevertheless, the weak equivalence shows that CFG power is sufficient for hierarchical structure analysis.

We have seen that our grammar is ambiguous. Hence it does not fall within deterministic subclasses of context free grammars such as LL and LR grammars. It is known that the worst case time complexity for parsing with general CFG rules is $O(n^3)$ where n is the length of the given sentence (that is, the number of words in the sentence). This implies that a parser will never take a time more than a constant times (n^3) amount of time to parse a sentence of length n. The constant in this complexity measure also involves a $\|G\|^2$ term where $\|G\|$ represents the number of grammar rules.

Traditionally, the complexity of parsing is expressed taking the length of the sentence, n, as a basis. The question that is asked is as the length of the input sentence keeps on increasing, how will the parsing time vary? However the length of a natural language sentence is not a good basis for complexity analysis at all. In compiling computer programming languages, a sentence would mean an entire source program, which typically runs into hundreds or thousands of lines. Here n can be vary large and it is highly variable. In natural language sentences, although there are no theoretical limits on the number of words in a sentence, sentences tend to be much smaller in practice, rarely crossing, say, 40 or 50 words. Complexity measures based on length of the input sentence tend to be misleading.

Secondly, and more importantly, the lengths of sentences in different languages vary significantly and systematically. A Telugu sentence typically has a much smaller number of words than the corresponding English sentence. This fact is a direct consequence of the fact that Telugu is a morphologically rich language and whatever English accomplishes by grouping several words together, Telugu achieves by appropriately inflecting a single word in many cases. The appropriate units of structure for describing both the hierarchical structure of clauses in a sentence and the functional structure of individual clauses are word groups, not individual words. For example, we can assert that every clause must have one verb group. Nothing can be said in general about the number of words that make up a verb group or the number of words in a clause. Number of groups in a sentence is a much better basis for complexity measure than the number of words.

In UCSG we obtain potential groups first and then carry out hierarchical structure analysis. Also, we have argued that noun groups do not contribute significantly for deciding the clause structure and we have used only verb groups and sentinels as a basis for hierarchical structure analysis. Thus for hierarchical structure analysis the n would just be the number of verb groups and sentinels in the input sentence. Thus the worst case time complexity of our hierarchical structure analyzer is $O(n'^3)$ where n' is the length of the verb group-sentinel sequence. In most cases this will be significantly smaller than the total number of words in the sentence. For example, if n' is one third the total number of words in the sentence, we get a twenty seven fold speed up compared to using the whole sentence for analysis. Further, we have only a very small set of rules in UCSG. The $||G||^2$ term is also very small. Thus hierarchical analysis in UCSG is very efficient.

It may be noted in comparison that ATN, LFG and TAG all go beyond CFG power. GPSG remains within CFG class but the expansion of the meta rules can make the $||G||^2$ term very large thereby making the whole system inefficient. In all these grammar formalisms there is no independent hierarchical structure analysis phase. All the advantages of the localizing effect of clause structure analysis are lost.

It is commonly taken for granted that sentences in natural languages can be arbitrarily long and complex. There can be an unlimited number of clauses nested in any arbitrary fashion. Hence it appears that at least the power of Type 2 or context free grammars is essential for dealing with the clause structure of natural language sentences. We will now argue that hierarchical structure analysis can also be done linear time, that is, with a worst case time complexity of n'.

It has been shown through psycholinguistic experiments that there is a limit to the depth of recursive nesting within a sentence beyond which human beings find it increasingly difficult to understand the sentence. See for example [23]. From a pragmatic point of view, we can assume that people normally speak so that they may be understood. It is not impossible that they may have other intentions but by and large, sentences that people use are intended to be understood. Consequently, people rarely if ever use sentences in which the depth of recursive nesting exceeds a certain threshold. This limit also happens to be quite small - three to four levels of nesting is about all that you find in natural language. Further, since this is a question about data, we can also carry out statistical analysis of large corpora of natural language text to practically determine the value of such a threshold. The actual value of such a threshold is not important for our arguments here. We do not attempt to determine or suggest a suitable value here. The fact that it is reasonable to place such limits on the depth of recursive nesting of clauses within a sentence is itself a very significant observation.

There is also a limit on the number of independent clauses that can be embedded within a given clause. A fundamental tenet of functional structure is that any sentence in any human language can be analyzed and partitioned into a small and fixed number of functional roles and each role can accommodate one and only one participant. Since the number of roles is fixed and each role can take at most one clause, there is a limit on the number of non-nested clauses within a sentence also.

Thus both the depth and branching factor of the clause structure trees are bounded. The main clause in a sentence has at most a finite and small number of clauses embedded in it, each of these can in turn have a finite and small number of clauses embedded within them, and there is also a limit to the depth of such nesting. If there are r functional roles that can take clauses and the depth of nesting is limited to d, there can be no more than $((r^d) - 1)/(r - 1)$ clauses in any given sentence. For example, if r is 6 and d is 4, we cannot have any sentence with more than 259 clauses in it including the matrix clause. In practice a vast majority of sentences will have much fewer clauses. The actual values of r and d are immaterial for our argument, the 'language' of clause structure is always a finite language. It is known that every finite language is regular and regular languages can be recognized using finite state machines in linear time. Hence *clause structure analysis can also be done in linear time, that too linear in the number of verb groups and sentinels only.*

2.4.7 Exceptional Constructs

We will now take up constructs which our H rules do not handle directly. The details of the exceptional constructs vary from language to language. It is difficult to provide general solutions. Also, we think the way we handle the rule is much more important than the way we handle the exceptional cases. In UCSG the emphasis is more on elegantly and efficiently handling the rule rather than on the specific details of how the exceptions are analyzed. We will only give here some of the possible approaches and say why some of these approaches may be better than the others. There are basically two kinds of exceptions - constructs that inherently lack sentinels and constructs in which sentinels and/or verbs may be optionally dropped out. Nominalized clauses are examples of the former. We shall call clauses in which sentinels and/or verbs have been optionally dropped out as *reduced clauses.* Let us now look at nominalized clauses and reduced clauses from the point of view of hierarchical structure analysis.

Nominalized Clauses

We have seen that nominalized clauses exhibit dual nature - they are clauses and noun groups at the same time. A consequence of their noun group nature is that there may be no overt sentinels. There are two possible approaches that we can take. As a first alternative, we can introduce more kinds of sentinels that can signal nominalized clauses and modify our H rules to include these clauses also for hierarchical structure analysis. Nominalized clauses vary quite a bit from language to language. In English for example, we may view the 'to' in infinitival verb groups as a sentinel. In some cases we will have to posit empty sentinels also. We may say, for instance, that a gerundial clause in English has an empty sentinel and a verb of the 'ing' form. We can then analyze the hierarchical structure including nominalized clauses as usual. However, this approach has several demerits. Not all infinitival clauses begin with 'to' in English (see example 69 below) and the word 'to' has other roles - it can also signal locative or dative case. Inclusion of empty sentinels and rules for nominalized clauses will make the H grammar more complex. The effective length of the input sentence for hierarchical analysis also increases. There will be a degradation of the nice properties of our H grammar.

69) Let him go
    (cf. Allow him to go)

Alternatively, we can simply treat these nominalized clauses as noun groups and ignore them completely during the hierarchical structure analysis. We gain the advantages of a very simple and universal grammar and very efficient hierarchical analysis. We simply defer the treatment of nominalized clauses to functional structure analysis. Entire nominalized clauses are anyway like noun groups in terms of functional structure. When nominalized verb groups are found during functional structure analysis, the functional structure of the corresponding clauses are also analyzed. We have noted that nominalized clauses tend to miss out on certain roles which will have to be borrowed from, and shared with, roles in the outer clauses. The functional structure grammar provides the necessary rules. We have used this latter approach in our implemented parsers.

Reduced clauses

Reduced clauses are clauses in which sentinels and/or parts of a verb group have been optionally dropped. Optionally dropped items are shown within parentheses in the following examples.

70) The book (which) the teacher asked us to read was difficult to follow.
71) I know (that) you are brilliant
72) The book (which is) lying on the table is mine.
73) The horse (which was) raced past the barn fell.

The crucial point to note is that the reduced forms of these sentences mean exactly the same thing as what their full counterparts mean. There is absolutely no difference in meaning. Native speakers seem to recognize that specific constituents have been optionally dropped, they automatically fill up the missing parts and analyze the sentence as if nothing were dropped. The construct '(which) the teacher asked us to read' *is* a relative clause and *we need to analyze it as a relative clause* even when the relative word 'which' is missing.

The fact that human beings tend to get led down the garden path in cases like 73) above is irrelevant here. Human beings process sentences left to right and under real time constraints. So they tend to make default assumptions which can go wrong at times. This is purely a matter of processing approaches and resource constraints. Sooner or later they do realize their mistake, they undo the faulty assumption and get the sentence right. As far as the meaning that the native speakers finally get is concerned, the omission of the optional constituents has no effect.

In UCSG, we believe that it is inappropriate to posit separate and special grammar rules to deal with purely syntactic variants of other constructs for which the grammar rules already exist. We should be able to parse the reduced sentences using the same grammar rules that handle the full form counterparts of these sentences. This makes the core grammar smaller, simpler, much more easier to write and more uniform across languages. We should be able to discern that something is missing, find out exactly what is missing and where exactly in the sentence it is missing. Then we can insert the missing items and process the sentence using the existing grammar rules. But how do we find out what exactly is missing and where?

Pragmatically speaking, speakers omit the optional items because they know that the listeners can easily identify that something is missing and put back the required things in the right place without much difficulty. In fact constituents become optional only if this is true. Constituents can be missing in only some restricted situations - situations in which the hearer can readily identify and fill up the missing items and understand the sentence as if nothing were missing. It would be ungrammatical to omit items that may cause real loss of information or introduce ambiguity. It should not be therefore difficult for a computational system also to determine when something is missing, where exactly in the sentence it is missing and which exact items are missing. We can then insert the right constituents at the right place and use the same existing grammar for parsing.

The details of which items can be optionally dropped and in what kinds of syntactic constructs vary from language to language. To some extent these details also depend on lexical ambiguities in specific words. Hence universal rules cannot be given. We will consider some examples here and provide some heuristic guidelines for detecting and inserting the missing constituents. Consider

74) I know that you are brilliant
75) I know you are brilliant
76) That you are brilliant is known to me
77) *you are brilliant is known to me
78) After the meeting was over all the boys went home
79) *the meeting was over all the boys went home
80) All the boys went home when the meeting was over.
81) *All the boys went home the meeting was over.
82) The boy who wrote the poem is brilliant
83) *The boy wrote the poem is brilliant
84) The boy whose book I had borrowed is brilliant
85) *The boy book I had borrowed is brilliant

In 75), the fact that the verb group 'know' permits a clausal object of the 'that clause' type and the fact that there is a clause after 'know' (we know this because we have already got the verb group 'are' in linear analysis phase) are sufficiently strong to indicate a missing 'that' sentinel. Without this sentinel the H grammar rules will fail and the insertion of 'that' gives a valid H-Structure. Examples 76) and 77) show that the sb 'that' cannot be missing in the subject position. Sentences 78) to 81) show that sbs indicating modifier clauses and indirect questions cannot be dropped.

As far as rls are concerned, observe that subject-relativizing rl 'who' and the determiner-rl 'whose' cannot be omitted (examples 82 to 85). The rls 'that', 'which', 'whom', 'where' and 'when' can be dropped. When an rl is dropped, we will typically get two noun groups in sequence - the noun group being modified by the relative clause and the subject of the relative clause. Since subject relativizers cannot be dropped, the relative clause must have a subject. In English, two noun groups appear in a sequence only in some limited situations such as in the double object constructions for bitransitive verb groups. Thus occurrence of consecutive noun groups in other contexts can be taken as an indication of a missing relative word. Which word is used is immaterial for hierarchical analysis.

In 72) and the garden-path sentence 73) above, we have nonfinite forms of verbs which in English have multiple uses. A verb in the 'ing' form can be used as part of a present tense continuous verb group, as a present participial or as a gerund. The past participial form can function as part of a verb group or as an adjectival modifier within noun groups. We need to consider all the possible uses. In both of these sentences, only one complete verb group is found and hence our H grammar would analyze these sentences as simple sentences. Sentence 72) is like 'something is mine' and sentence 73) is like 'something fell'. During functional structure analysis we can insert the missing relativizer and an auxiliary verb of the 'BE' type and analyze as relative clauses.

Alternatively, we can hypothesize missing constituents at the stage of linear structure analysis itself. Whenever we encounter a noun group followed by a present or past participial verb that forms neither a part of a verb group nor a part of a noun group, we insert a relative word and an auxiliary verb of the 'BE' type between the two. Then hierarchical structure analysis and functional structure analysis can go on as usual. Note that sentence 73) will no longer be a garden-path sentence. Our aim here is not to build a faithful model of human language processing. We do not expect our syntactic system to behave exactly like humans do. We do not make any cognitive claims for our theory. We instead want a grammar formalism that helps to write computational grammars easily and to parse natural language sentence efficiently.

In general, dropping of optional items would cause the hierarchical structure analysis or the functional structure analysis to fail on the given sentence. This can be taken as an indication of a possible dropping of sentinels and/or auxiliary verbs. The type and location of failure indicates what types of items need to be inserted and the place in the sentence where these items need to be inserted. In many cases we can even obtain possible types of missing items and their locations during L analysis itself. Knowing the possible missing items and their locations helps to quickly and easily patch up the given sentence and obtain the correct analysis. Partial analyses already obtained can be reused directly or with the required modifications. This in general would be a better strategy compared to directly writing grammar rules to handle all possible combinations of optionally dropped items. That would make the grammar unnecessarily large and complex, the parser very inefficient, and most importantly, the inherent generalization in terms of meaning between the full form and the reduced form would be lost. Also, the nature of reduced forms vary from language to language and directly writing grammar rules for each case would also render the grammar less uniform across languages.

2.4.8 Concluding Remarks

The uniqueness of UCSG lies in positing an independent H level dealing with the hierarchical structure of clauses in a sentence. In UCSG hierarchical structure analysis is done before and without the application of functional structure constraints. Knowing the structure of clauses in a sentence permits localizing the functional structure analysis to essentially one clause at a time. In a sentence with n clauses, this gives the effect of analyzing n simple sentences rather than one complex sentence with n clauses in it. As a result functional structure analysis is greatly simplified.

CFGs are ideal only when both linear *and* hierarchical structure are significant and functional dependencies are not relevant. H-Structure has just these properties. As far as H-Structure is concerned these properties hold equally well for positional languages and relatively free word order languages. CFG power is required and just sufficient for dealing with H structure. Hierarchical structure analysis can be carried out using only a handful of CFG rules with a worst case time complexity of $O(n'^3)$ where n' is the length of the verb group-sentinel sequence. Further, we have argued that hierarchical structure analysis can also be reduced to linear time complexity if we make certain assumptions justifiable on psycholinguistic and statistical grounds. Both the depth and the branching factor in clause structure trees are bounded making the language of clause structure finite and hence regular language.

The H grammar rules are simple and language invariant. Both positional languages and relatively free word order languages follow the same grammar with at most parametric variations. Since the clauses that enter into cross serial nesting in Dutch are all nominalized and nominalized clauses need not enter into the H structure analysis, even Dutch can be parsed by the same H grammar. In the next section we show that identifying the clause structure first also leads to an elegant solution to the problems of movement and long distance dependencies.

There are, however, exceptional constructions which are not directly handled by the grammar rules we have given. Since the details of exceptional constructions vary from language to language, no general solutions can be given. If the rule is handled efficiently, the system is likely to be efficient as a whole is also, since exceptions are likely to be less frequent than the rule in practice. We believe that it is better to handle syntactic variants of regular constructs by noticing the nature and location of the variation and patching up rather than by positing separate rules for all possible constructs. We have given some guidelines for doing this by taking examples from English.

2.5 Functional Structure

We start this section by specifying what exactly we mean by functional structure. We then propose a set of functional roles and say why we think this forms a good set of roles. After that we will get into the details of the grammar of functional structure and discuss the relative significance of the different kinds of functional structure constraints. Following a discussion on displaced, shared and missing participant constructions and long distance dependencies, we propose a *work from whole to part* strategy for dealing with movement and long distance dependencies. We propose that the functional structure analysis be done clause by clause starting from the main clause and going into the nested clauses recursively. We pass on information and expectations about displaced, shared and missing participants so that functional structure analysis becomes almost completely local to a clause. There will no longer be any long distance dependencies. Finally, we show that analyzing the functional structure clause by clause makes the functional structure analyzer as well as the entire parsing process very efficient in UCSG.

Of all the issues taken up in this section, analyzing functional structure clause by clause and the *work from whole to part strategy* are the most crucial for UCSG. The exact set of functional roles we use or the details of how exactly we assign the roles to the various participants are less important. Any approach that is computationally viable and efficient can be used. These latter issues do not constitute any major claims of UCSG.

Let us start by understanding what exactly we mean by functional structure. Suppose I am now narrating a story that I had written several years ago. One question that immediately arises is what exactly was stored in my mind over all these years that enables me to reconstruct the story now? It is quite obvious that entire sentences are not stored. In all probability, I would now be using entirely different words, phrases and even sentence structure. But I must somehow remember the events, the sequence of events, the interplay between various events and so on. Remembering an event means, in the least, remembering what happened, to whom, where, when, why, and so on. An NLP system is required to produce meaning representations at least at this level of abstraction. A true meaning representation should use language independent semantic primitives since it is possible to use not only different words and expressions but also a different language to retell a story. However, a syntactic module can only produce structural descriptions, not meaning representations. What do we really expect from a syntactic system then?

We believe that it is possible within a syntactic system to assign functional roles to the word groups at a grammatical level of abstraction. Each predication says something about one event, action or state. Each clause in a sentence corresponds to one predication. A clause includes a verb group that indicates the action, event or state and noun groups that indicate the various participants in that action, event or state description. Each noun group has a function - it plays one role in the predication. Noun groups may take on functional roles such as doer, done-to, place, time etc. The functional structure of a sentence depicts the assignment of functional roles to the various groups in each of the clauses in a sentence. The participant roles of each predication would be filled by the actual words used in the input sentence, not by universal semantic primitives of any kind. Hence the functional structure is only a structural description, not really meaning representation. We believe that a good syntactic system should not only analyze the linear and hierarchical structure of sentences but also the functional structure.

In UCSG we assign roles to all the participants, whether they are mandatory or purely optional participants. We therefore prefer the more general term 'functional role' for more common terms 'case role' and 'theta role'. We call this aspect of structure 'functional structure' or F-Structure for brevity. The F-Structure of a complete sentence also shows the interdependencies between the various clauses in the given sentence. Each clause except the main clause is related to another clause in one of the two ways - it takes up a particular functional role in the enclosing clause, or it acts as a modifier of one such participant. Thus the hierarchical structure of clauses in the sentence are also depicted clearly. Also, all the word groups are depicted retaining the linear order of words within the groups. So linear structure is also indicated. Thus the F-Structure of UCSG is a complete structural description that clearly shows linear, hierarchical and functional structure inherent in the input sentence.

The F-Structure is basically a tree in which each of the nodes shows the functional structure of one clause. The functional structure of each clause is a set of (word group-functional role) pairs. The F-Structure may also indicate optionally dropped word groups and groups which are shared between two or more clauses. Sharing of participants turns the F-Structure into a graph. Examples of how the F-Structure is displayed on the computer screen were seen in the first chapter. We will see the internal representations of these later on in this section. We will now start with the issues of how many and which functional roles to use for functional structure analysis.

2.5.1 Functional Roles

The basic principles behind functional or case roles are not new at all. ‘karaka’ roles in the Paninian system of grammar of Sanskrit, Fillmore’s case system and all the other case systems that have been proposed since Fillmore, as well as the adaptations of case concepts in more recent theories like the Government and Binding theory, all hinge around a few fundamental ideas. Firstly, case theories assert that there is a small set of universal and semantically oriented case roles into which all kinds of sentences in any human language can be analyzed. The set of case roles is required to be small to achieve a high degree of generalization across different syntactic constructions and across different human languages. Secondly, each case role can be filled by one and only one constituent except when two or more constituents are conjoined by explicit coordinate conjunctions forming a single compound constituent. Thirdly there are certain completeness and coherence criteria to be satisfied at the level of a clause. All the required roles should be present and no role that is disallowed can appear. The participants present must make the entire predication syntactically and semantically coherent. No part of the sentence must be left out and no part can play more than one role unless syntax permits and demands. These completeness and consistency constraints are commonly expressed in terms of subcategorization frames, selectional restrictions and agreement requirements.

While the basic tenets are well known, open questions remain in all case theories. How many case roles should there be? Which ones? How do we define these roles precisely? How do we specify constraints for mapping the various word groups onto the case roles? Different case systems differ only in terms of the particular answers they provide for these questions. There is no one case system which is universally accepted to be the best.

In UCSG, we have aimed at developing a small set of universal functional roles. The functional roles that we use are intended to be semantically oriented, universal, and language independent. We have related our case roles to questions that one might ask about a sentence. We have found that question-answer view helps a lot not only for designing the case system but at all levels of theory and implementation in NLP

[34]. The case system we use here is a refinement of the system we had developed in [35]. Our case system is influenced by considerations of computational viability and efficiency. The case role definitions are semantic in nature but we require that it must be possible to specify role assignments in almost purely syntactic terms. It should be possible to assign such case roles automatically and without requiring human intuitions and judgements. In UCSG we depend primarily on syntactic information and employ only a bare minimum of semantic features that are essential for proper role assignment. Our set of cases is smaller than many others proposed earlier. Smaller the number of cases, higher the level of abstraction and generalization. Thus our case roles are often generalizations of two or more finer grained case roles used in other case systems.

We will now look at our case roles and make some remarks on how they compare with the case roles defined in different case systems. We will be particularly concerned with computational viability issues. It must be emphasized that while the case roles themselves are semantic in nature, we need to be able to provide computationally viable and efficient algorithms for identifying these case roles in a given sentence using primarily only syntactic information. We also provide examples of default rules for role assignment. We have eight roles the first one being

1. SUBJ :

Every clause has a verb group that specifies an action or a state. Any action requires a doer of that action and any state description requires somebody or something whose state is being described. We denote this case of the doer, or the person or thing being described, by the role SUBJ. In terms of names of the roles we have found it better to use existing terms despite their differing and sometimes confusing usage in different theories, than to coin new terminology of our own. We have used the traditional term subject denoted by 'SUBJ' for this case role.

The SUBJ role filler can be thought of as the answer to the question WHO or $\text{WHAT}_{nominative}$. It must be noted that in English we use the word 'who' if the answer is known to be a human being and we use 'what' when we know that the reply refers to a non-human entity or when nothing is known. It should also be noted that in

English we use the word 'what' in both nominative and accusative sense. English does not show surface cases through morphological inflections and one has to be careful to note the difference between the nominative and accusative usage. Here what is implied is the nominative sense. The functional role being questioned by the words 'who' or 'what' is not be confused with the English words 'who' or 'what'. Look at the following examples:

86) Rama killed Ravana
(cf. Who killed Ravana?)
SUBJ : Rama

87) Hanuman was very brave
(cf. Who was very brave?)
SUBJ : Hanuman

88) Cigarette smoking is injurious to health
(cf. What is injurious to health?)
SUBJ : Cigarette smoking

89) That the company sold defective computers annoyed everyone
(cf. What annoyed everyone?)
SUBJ : That the company sold defective computers

90) To err is human
(cf. What is human?)
SUBJ : To err

Pragmatically, SUBJ is essential. Any action requires a doer of that action and any state description requires somebody or something whose state is being described. Syntactically also this happens to be the most compulsory role. There are only a few kinds of sentences where the SUBJ can be missing and in all these cases there will be sufficient syntactic information within the sentence to clearly identify the understood SUBJ. Examples are imperative sentences with the implied SUBJ 'you' and Telugu sentences like 'ippuDee vaccaanu' where the SUBJ is clear from the person-number suffix on the verb group itself.

The subcategorization frame of the verb group tells us what syntactic types of constituents can take the SUBJ role. Thus 'surprise' can take a 'that clause' while 'eat' can take only a noun group. Groups of the appropriate type that also satisfy the surface case marking requirements form the potential candidates for the SUBJ role. In English the SUBJ appears to the left of the verb group. In Telugu, the SUBJ is in the nominative case by default.

The nearest case roles in other cases systems are AGENT, EXPERIENCER and KARTA. Fillmore has defined AGENT as 'the case of the typically animate perceived instigator of the action identified by the verb' [16]. He adds a foot note which reads

> "The escape qualification 'typically' expresses my awareness that contexts which I will say require AGENTs are sometimes occupied by 'inanimate' nouns like robot or 'human institution' nouns like nation. Since I know of no way of dealing with these matters at the moment, I shall just assume for all AGENTs that they are 'animate' ".

How does a computer understand 'typically animate'? How does a computer tell what is 'perceived' or what is really meant by instigation? In a computational framework, we need to specify how the AGENT can be automatically and mechanically identified without calling for human judgements, intuitions and common sense. Can we give an algorithm to determine the 'typically animate perceived instigator of the action identified by the verb'?

In his classical example 'John opened the door with a key', Fillmore says that 'John' is the AGENT. In the sentence 'The key opened the door', Fillmore argues that the key cannot on its own open the door and hence there is an unspecified AGENT - somebody who used the key to open the door. If one accepts the view that entities not overtly present in the utterance can be, and should also be considered for case role assignment, it can be argued that even in the case of the first example, it could very well be the fact that somebody else is forcing John to open the door and this somebody, being now more autonomous, independent and acting more by will than by force, could be ‘perceived’ as the ‘instigator’ and hence the AGENT. One may go to the extent of saying that God is ‘The AGENT’ for everything and man is only an instrument. Who is really the AGENT and who is doing by compulsion, out of ignorance or by accident? These are certainly not matters of syntax. We cannot assign roles to groups which do not exist. We argue that *every role in a clause must have an overt representation in the sentence*. We can hypothesize implicit roles only in limited situations where syntax permits items to be omitted, as in the case of the implied ‘you’ in ‘come in’.

The notion of agency has other problems. Consider the following sentence and its passive form.

91) His uncle frightened the boy
92) The boy was frightened by his uncle

In the active sentence, ‘his uncle’ would surely be taken as the AGENT whereas in the passive form, ‘his uncle’ may or may not be the AGENT - it depends on whether the uncle intentionally frightened the boy or the boy got frightened on his own. It is also very much possible that the uncle unintentionally, unknowingly or accidentally did something that frightened the boy. If the uncle burst a balloon and the boy got frightened by it should we say that the balloon frightened the boy or that the bursting of the balloon frightened the boy or that the bursting of the balloon by his uncle frightened the boy? Is it also not possible that the frightening was partly due to uncle’s intentional actions and partly due to the boy’s own apprehensions? The notion of AGENT is far beyond syntactic computation.

It is no big surprise that passivization affects agentiveness. Pragmatically speaking, we use the passive construction *so that* we can de-emphasize the AGENT, *only* *when* we do not want the AGENT to be highlighted or even mentioned.

Some theories attempt to capture the 'deep subjects' of passive voice constructions by considering the active voice counterpart of the given sentence. We have two kinds of objections for this approach. Firstly, there is no basis for taking either the active voice or the passive voice as more basic than the other. While active voice is the default in English, passive is the rule in Sanskrit. Should we transform passive voice sentences into active or the other way round? Also, given the sentence 92) we can ask the question 'who was frightened by his uncle?' but 'who/what frightened the boy ?' is not a question that can be answered using syntax alone. Questions that involve voice transformation may in general beg large amounts of semantic and pragmatic knowledge. Our aim here is syntactic analysis which we view as a step towards understanding the meaning. It is therefore absurd to take steps that can potentially change or distort the meaning. In UCSG we will only analyze the SUBJ of the sentence as given. Passive sentences will remain passive and active sentences will remain active.

Panini's karaka theory is better. Entities not explicitly present in the sentence are never considered. Among the entities mentioned overtly, the most independent participant would be taken as the KARTA (literally 'the doer', hence similar to the notion of AGENT). This definition too is problematic, however, since what is required is a measure of relative independence amongst all the entities present in the sentence. Should we provide statically in the lexicon the relative independence of all possible entities or can we give algorithms to compute 'independence'?. It has been argued that in practice, approximating the notion of independence with a few static semantic features such as animacy and humannes may prove to be sufficient in a majority of cases. Is it not an unwarranted complication to define something that is not always realizable in practice and then to approximate it with something simpler? If animacy and humanness are sufficient why not define the case roles directly in terms of these rather than in more semantic and unrealizable terms?

Many case systems have made distinctions between AGENT and EXPERIENCER. See [14] for example. The distinction between the two is in terms of such aspects as the degree of autonomy, independence intentionality or volition, none of which are easy to 'compute'. consider

93) Water flows from a higher level to a lower level

94) Wood floats on water

Does water flow on its own or does gravity force water to flow and water only experiences the force of gravity as it flows? The weight of the wood tries to sink the wood and the buoyancy force in the water tries to push the wood up. There is a balance and the wood floats. Who then is the AGENT and who is the EXPERIENCER? Paninian theory attempts to answer such questions by hypothesizing that a given action can be broken into a hierarchy of sub-actions and each such sub-action will have its own KARTA. Floating of wood is an action that includes sub-actions such as gravity pulling the wood downwards and buoyancy pushing the wood upwards. While this model may have a place in semantic theories, this is of no use to a syntactic system where we are only interested in assigning functional roles to the overt noun groups in a clause with a given overt verb group in it. We have neither the sub-actions nor the participants at those levels with us.

The distinctions between AGENT and EXPERIENCER cannot be captured easily within a computational system of syntax. We should instead merge the AGENT and EXPERIENCER into a single more abstract role. Such a generalized role is real and valid, linguistically and psychologically. Consider the following pairs

95) Krishna killed Radha

96) Who killed Radha?

97) Radha died

98) Who died?

Are not both of these questions grammatical, meaningful and acceptable in real life? Yet we have used the same expression 'who' to question the AGENT as well as the EXPERIENCER. Our case role SUBJ is a generalization of the AGENT and EXPERIENCER, the role expressed by the question word 'who'. 'Krishna' is the SUBJ of 95) above and Radha is the SUBJ in 97).

Notions such as volition and will, independence and autonomy, doing something deliberately or accidentally and unknowingly are beyond syntax. These qualities cannot be quantified and measured using syntactic information. We need computable and syntactic definitions. This requires making generalizations. We believe that our 'SUBJ' is at the right level of abstraction.

2. OBJ :

While the SUBJ indicates the doer or the experience, the OBJ role indicates the done-to, the person or thing that is acted upon. Obviously OBJ occurs only with transitive verbs. The OBJ role filler will be the same as the answer to the WHOM or $\text{WHAT}_{accusative}$ questions.

99) Rama killed Ravana
(cf. Whom did Rama kill?)
OBJ : Ravana

100) Rama ate three apples and two oranges
(cf. What did Rama eat?)
OBJ : three apples and two oranges

101) Rama knew that the golden deer that Sita fell for was not real
(cf. What did Rama know?)
OBJ : that the golden deer that Sita fell for was not real

102) Rama knew who lifted Sita
(cf. What did Rama know?)
OBJ : who lifted Sita

The subcategorization frame tells whether the OBJ role is required and if so, it also specifies the syntactic types of constituents that can take up the OBJ role. The OBJ role will be disallowed in other cases. In English OBJ occurs to the right of the verb group. In Telugu it occurs in the accusative case by default.

We will provide our remarks on comparable case roles in other systems after we define two other cases : SUBJ_QUAL and OBJ2.

3. SUBJ_QUAL :

Consider the sentences

103) Hanuman was very brave

104) Rama became a doctor

The verbs 'be' and 'become' are not transitive. We cannot ask a WHOM or $\mathrm{WHAT}_{accusative}$ question on the assertions above. The WHAT in the question 'WHAT did RAMA become?' is not in the accusative. There are really no OBJs in these sentences. These verbs are not transitive yet they do not specify any action or state completely on their own. The following sentences are incomplete.

105) *Hanuman was

106) *Rama became

Traditional grammars classified such verbs under the head 'verbs of incomplete predication' which is exactly what they are. The verb 'hitting' indicates some action by itself. On the other hand, 'being tall', 'being a nice person' are complete state descriptions but 'being' (not in the sense of existing) does not by itself indicate any action or state completely. Verbs like 'be' and 'become' are verbs of incomplete predication. These verbs always require an adjective phrase or a noun phrase to specify the state completely.

We define a role called SUBJ_QUAL. In an abstract sense, SUBJ_QUAL can be taken as the answer to the question ‘WHAT$_{nominative}$ SUBJ vg?’ where the vg indicates the verb of incomplete predication in question. As the name indicates, SUBJ_QUAL qualifies the SUBJ rather than indicating a new entity. In the above examples, Hanuman himself is described as being brave and there is none other than Rama implied when we say ‘a doctor’. Note that SUBJ and OBJ normally indicate separate entities unless they are coindexed through the use of a reflexive pronoun as in

107) Rama killed himself

Note also that OBJ and SUBJ_QUAL are mutually exclusive - they cannot occur together in a single clause. Distinguishing between OBJ and SUBJ_QUAL is useful and well within the scope of syntax. The distinction simply follows from the type of the verb. The subcategorization frame for such verbs requires that there be a SUBJ_QUAL role and it specifies whether that role can be taken up by a noun group or an adjectival group. In English SUBJ_QUAL appears in the same position as OBJ. In Telugu, it appears in the nominative case by default.

4. OBJ2 :

There are bitransitive verbs, verbs that require two objects. One of these objects will fill the OBJ role as we saw above. The other secondary object indicates things such as donor, recipient and beneficiary. We will denote this role by the term OBJ2. The filler of the OBJ2 role will be the answer to questions such as TO WHOM, FROM WHOM and FOR WHOM.

108) Dushyanta gave a ring to Shakuntala
(cf. To whom did Dushyanta give a ring?)
OBJ2 : to Shakuntala

109) Venkateshwara had borrowed a lot of money from Kubera
(cf. From whom had Venkateshwara borrowed money?)
OBJ2 : from Kubera

110) Krishna created an endless saree for Droupadi
(cf. For whom did Krishna create an endless saree?)
OBJ2 : for Droupadi

Observe that the slot fillers in these examples are prepositional phrases. As already stated we view prepositional phrases as equivalent to noun phrases except for overt case marking through the use of the preposition. We do not propose separate case roles such as DONOR, RECIPIENT or BENEFICIARY. We think these are really minor variations of the same role and the distinctions among them cannot always be done using syntactic information alone. A single case role seems to be a good generalization.

Now let us look at case roles similar to our OBJ, SUBJ_QUAL and OBJ2 in other case systems. Fillmore uses three cases named OBJECTIVE, DATIVE and FACTITIVE that may be the closest equivalents to the roles we have defined. Fillmore defines DATIVE as 'the case of the animate being affected by the state or action identified by the verb', FACTITIVE as 'the case of the object or being resulting from the action or state identified by the verb, or understood as part of the meaning of a verb' and OBJECTIVE as 'the semantically most neutral case, the case of anything representable by a noun whose role in the action or state identified by the verb is identified by the semantic interpretation of the verb itself'. Again qualifications like 'affected by' and 'resulting from' are completely useless for building a computational theory. How should our computer know what is semantically the most neutral? A Computational grammar formalism should be such that the grammar writers can easily specify the grammatical constraints and in as uniform and undebatable way as possible and the computer should then be able to mechanically and automatically apply these grammatical constraints to perform syntactic analysis without calling for human intuitions and judgements.

Paninian theory defines the case KARMA in a completely different way. One definition takes KARMA as 'the thing most desired by the KARTA'. Firstly, how do we give an algorithm for identifying what is most desired by the KARTA? Secondly, Paninians would call ‘home’ in

111) Krishna went home

a KARMA since that is the thing most desired by the KARTA Krishna. We have always found that meanings are best judged in terms of questions and answers. Everybody would agree that ‘home’ is the *place* where Krishna went. ‘Home’ cannot be assigned the OBJ role. While the Paninian system may be internally consistent, the case role assignments are not in line with what common people think. It is simply confusing to people because there are many other examples where KARMA would coincide with the role we have called OBJ.

Our definitions are question-answer oriented and thus semantic in nature. We can depend on the intuition of people for understanding the nature of these roles. There is no need for elaborating on the semantics of these roles or attempting to provide precise definitions. Such an attempt can only make things more confusing. Questions and answers also do not always appear intuitive and natural. Yet we believe the question-answer view is much more intuitive in comparison with other approaches.

5. SPACE :

There are a few verbs like ‘put’ that require a place specifier to make the sentence complete. Sentence 112) below sounds incomplete while 113) appears complete.

112) *I put the book

113) I put the book on the shelf

We have called our locative case role as ‘SPACE’. This can be taken as the answer to the WHERE question taken in its broadest sense. We have not made finer distinctions such as a point in space or a region, origins and destinations, direction, etc. All of these relate to space, all these aspects are merged into the SPACE role. The verb ‘put’ requires SPACE but for ‘eat’ SPACE is purely optional. SPACE is optional for a vast majority of verbs.

There are other cases which are always optional. Since they are never mandatory, most case systems do not treat them as cases at all. For example they do not consider time as a case role. They make a distinction between roles which may be required for at least some verbs and roles which are always optional. Terms like Inner Case / Outer Case, Arguments / Non-Arguments, A-position / $\bar{A}$-position have been employed in different theories for this purpose.

One should not mix up completeness and grammaticality. Completeness is but one essential requirement for a sentence to be grammatical. Consider

114) *Krishna slept the baby

Here the ungrammaticality is due to having more than complete information, information which should not have been there. Therefore in general a subcategorization frame should clearly indicate not only which roles are mandatory, but also which ones are optional and which ones are prohibited. Case grammars have emphasized the completeness aspect, calling only the roles which are essential to make complete sense as cases. Semantically, optional cases make as much a contribution to the full meaning of the sentence as other cases do. We leave completeness issue to subcategorization frames and call all the cases as functional roles, whether they are essential or optional.

We now go on to present our remaining cases. Our optional case roles include

6. TIME 7. ADVERBS and 8. CAUSE_LINK.

These correspond to the WHEN, HOW and WHY questions taken in their broadest senses. A specific point of time, duration, frequency etc. are all packed into the TIME case role. We include manner adverbs as well as sentential adverbs under the ADVERBs role. The CAUSE_LINK role includes indicators of cause, reason, purpose, etc.

In summary, we have eight case roles named SUBJ, OBJ, OBJ2, SUBJ_QUAL, SPACE, TIME, CAUSE_LINK and ADVERBS. Our case roles are more abstract, more general and hence overloaded. Our aim has been to define a small number of case roles such that case role assignments can be made easily using mostly syntactic information only.

Observe that we have not included an INSTRUMENT case role. Other cases systems have probably been influenced by the fact that some languages exhibit overt surface case marking for the instrument case. It is not practically possible to determine when some phrase is an instrument and when it is something else. The following examples from Winograd [52] should help to appreciate the difficulty.

115) He built it with mahogany
116) He built it with a sledge hammer
117) He built it with an experienced carpenter
118) He built it with the other furniture
119) He built it with difficulty
120) He built it with no purpose in mind

We may say that 'with a sledge hammer' has the force of 'using a sledge hammer' and hence this is the case of an instrument. Is the experienced carpenter not 'instrumental' in making whatever is being made? Is mahogany also an instrument? Or should we create a new case role called, say, MATERIAL? To do something we may use instruments, raw materials, effort, time, thinking. Are all these separate cases?

The question is when and up to what extent should we keep making distinctions and when should we simply be content with a generalization. We can keep on diving the world to any extent. Whichever way we divide, we will not be fully satisfied since there is nothing like a perfect classification of the whole world. We have taken the view that a more generalized and more abstract set of cases would be more practicable.

2.5.2 The nature of Functional Structure Constraints

The F part of the grammar provides constraints for assignment of functional roles to the groups already obtained. The constraints for role assignment take a variety of forms including linear position constraints, morphological inflections and pre/postpositions, subcategorization and selectional restrictions and grammatical agreement requirements. We call all these types of constraints as functional structure constraints or simply F constraints.

The degree of significance attached to different types of functional constraints varies quite a bit from language to language. Not all constraints on functional structure are equally useful. A thorough understanding of the degree of significance of the various types of functional constraints is essential for building a simple and efficient theory of syntactic analysis of natural language sentences.

Some languages attach a great degree of importance to linear order of groups while others do equally well with very little constraint on linear order. The positional constraints of English are practically absent in Telugu and other relatively free word order languages. These languages compensate for this loss of constraint by imposing alternative constraints in the form of morphological inflections and pre/postpositions. This indicates that there is something common between these various types of constraints.

Let us consider the different kinds of F constraints one by one. Consider the sentence

121) *The table drank the elephant

This sentence is bad because selectional restrictions are violated. Tables cannot drink and elephants cannot be devoured like water. Selectional restrictions are direct reflections of the reality - they mirror the truths in the real world. These constraints are therefore universal. The fact that one cannot drink elephants is not dependent on any specific language.

Of course the nearest equivalent words in different languages do not always possess identical sets of senses and as a consequence there can be language to language variation. In English we have separate words for eating, drinking and smoking. In Hindi there are two words, one for eating and another for drinking and smoking. In

Bangla the same word can be used in all the three senses. Thus 'eating' cigarette is grammatical in Bangla while it is unacceptable in English. The difference is due to the fact that word senses are partitioned differently in these languages and not because of any real or perceived differences of the real world. Selectional restrictions are by and large universal.

Yet we argue that selectional restrictions should not be taken too seriously. How often do you find people speaking sentences like the above in their normal communications? Selectional restrictions are so basic to the world we live in, as also to our language, that both native speakers and foreign language speakers rarely violate these restrictions. So in practice, these constraints are not very useful in setting apart good sentences from the bad. Do they help us in selecting the correct structure amongst a set of possible structures at least? Consider the sentence

122) Radha had a little lamb

How do we know that we are using the word 'has' in this sentence in the sense of possessing? Compare with

123) Radha had three slices of bread (for breakfast)

124) (Radha was expecting.) Radha had a baby boy.

The real world knowledge that one eats bread and one gives birth to a baby helps us to identify the correct or default word senses. Selectional restrictions, which are simply representations of these real world facts, are also thus useful for word sense disambiguation.

It is very important to notice however, that word sense differences do not necessarily change the F-Structures. Whether Radha possessed or ate or gave birth to something, Radha is the SUBJ and that something is the OBJ. There is thus only one F-Structure for sentence 122. Selectional restrictions are not very useful for determining the correct functional structure which only show the mappings from word groups to functional roles. In fact the F-Structures obtained can be used for word sense disambiguation. In UCSG we recommend that selectional restrictions in the form of semantic features be incorporated into the syntactic system only to the limited extent they help to distinguish the correct F-Structure from others. In UCSG F-Structures are the final and complete structural descriptions. We have found that we can produce the correct analysis and also limit the alternative F-Structures produced for a given sentence to very small numbers using only a bare minimum of semantic features. As far as syntax is concerned, we do not need large or unbounded amounts of semantic or world knowledge.

Agreement rules are purely grammatical in nature and hence vary substantially from language to language. People naturally tend to make mistakes in agreement quite often. However, agreement requirements are not directly related to real world facts and hence sentences with agreement mistakes are normally easily understood by the listeners. In real life applications, even computational models will be required to accept and correctly handle sentences with such mistakes. Agreement rarely if ever decides the structure of a sentence. Agreement cannot form a strong basis for deciding which role to assign to which word group. These are weak constraints more useful for verification than for decision making.

On the other hand, constraints on the syntactic types of various roles are much more useful. Some verbs take noun groups as OBJ, some take indirect questions, some take 'that clauses' etc. Knowing these types can substantially reduce the search. We also need to know which roles are mandatory, which roles are prohibited and which ones may appear optionally. Subcategorization frames encode all these kinds of information and thus play a very important role in F-Structure analysis.

In UCSG we view linear order constraints as well as constraints on morphological inflections and pre/postpositions as different manifestations of one and the same phenomena which we call *surface case marking*. The primary purpose of surface case marking is, according to us, to indicate possible functional role assignments. Surface case markers are all thus extremely important for F-Structure analysis.

Summarizing, UCSG depends mainly on subcategorization frames and surface case markers for functional role assignment. Selectional restrictions are used only if and where required to distinguish between the correct F-Structure from other possible ones. Grammatical agreement is used only for verification after the F-Structure is obtained.

2.5.3 Displaced, Shared and Missing roles

The subcategorization frames specify which participant roles are essential, optional and prohibited. These constraints are directly applicable to a simple sentence but in complex sentences, these constraints must be applied carefully. In Complex sentences, roles can be displaced, shared between two or more clauses or they may be completely missing. Relative clauses, indirect questions and infinitival clauses are examples of such syntactic constructions. To that extent the functional structures of clauses are dependent on one another. So, it seems that functional structure analysis cannot be made completely local to the respective clauses.

However, these interdependencies amongst clauses are limited. A proper understanding of the clause-to-clause dependencies helps. We will show that all the inter clause dependencies are related to the hierarchical structure of clauses in a sentence. We will see that each clause can be analyzed essentially in isolation if we work from the main clause down the clause hierarchy, and if information and expectations about displaced, missing and shared constituents is passed down as a parameter for the analysis of each clause.

We believe that the source of these displaced, shared and missing participant constructions lies not in syntax but in semantics and pragmatics. That is why these phenomena are universally found across human languages. This realization helps us to visualize grammars and parsers correctly. Let us look at the syntax of such constructions from this point of view.

We use relative clauses to modify noun groups. A relative clause modifies a noun group by specifying a predication in which that noun group plays some role. Hence every relative clause will have a participant role identical to the noun group being modified. Pragmatically, we prefer saying something once rather than twice. Syntax has exploited this and transformed this preference into a requirement - every relative clause must have a missing role. In order to analyze a relative clause, therefore, we need to view the noun group being modified as one of the participants in the relative clause also, - this noun group is shared between two clauses. The missing participant can be simply borrowed from the noun group that the relative clause is modifying. Of course the relative clause may itself have other clauses within it and hence there can be one or more clauses placed linearly between the shared noun group and the place it fits in its modifying relative clause. Thus there can be 'long distance dependencies'. We take up the issue of long distance dependencies in the next section and provide the required sharing rules.

In a relative clause the sentinel itself provides useful clues about which role is missing. Thus 'when' relative clause must have the TIME role missing, 'who(m)' relative clause must have the SUBJ (OBJ) role missing, and so on. As far as hierarchical structure analysis is concerned, knowing that a word is an rl was sufficient but for functional structure analysis it helps to know which exact rl is used. Grammatical categories provide the required generalization but there is useful information in specific words and this information must be exploited appropriately.

When there are ambiguities left over, other F-Structure constraints join hands to disambiguate. Since each role can be filled only once, roles which are already filled need not be considered. The noun group being modified by the relative clause must fit in the missing role in terms of syntactic constraints like subcategorization frame requirements and selectional restrictions. Taken together, the F constraints are sufficient to identify exactly which role is missing in a relative clause. The rl 'that' can signal both subject and object relativization. In 125) it can only be subject relativization and in 126) it can only be object relativization because the other roles are already filled. Sentences 127) and 128) are bad because there is no free role. Since 'that' allows both kinds of relativization and 'dress' can be used both transitively and intransitively, 129) and 130) are both valid.

125) the boy that read the book

126) the book that the boy read

127) *the boy that the girl read the book

128) *the book that the girl read the Bible

129) the baby that the boy dressed

130) the fact that the boy dressed

Indirect questions also have a missing participant but this is not a shared role, it cannot be borrowed from elsewhere. As far as the semantics of the indirect question itself is considered, the missing role may be replaced with the answer to the corresponding direct question. Consider the examples

131) Krishna ate what Radha cooked

132) Krishna guessed what Radha cooked

In both 131) and 132) above, the answer to the direct question ‘what did Radha cook’, let us say, rice, can be inserted in the missing position to make a complete and grammatical clause ‘Radha cooked rice’. Note however that semantically, these two sentences are not similar. In 131) Krishna ate the very same thing that Radha cooked, namely rice. The sentence 131) has the same force as ‘Krishna ate the thing which Radha cooked’. This indirect question is more like a relative clause. So the term ‘indirect question’ is not always appropriate but we will continue to use this term since it has already become well established. In 132) Krishna did not ‘guess rice’ but he ‘guessed that Radha cooked rice’. 133) should make this distinction even more clear, since both the interpretations seem possible.

133) Krishna knows what Radha knows

Despite this semantic difference, our F-Structures will not show any difference in the two cases. Our aim is only to capture the syntactic structure of sentences and, syntactically, these sentences will be treated identically. In all the above cases the matrix clause would have the indirect question in its OBJ slot and the indirect question would have an empty OBJ. We leave it to a subsequent semantic analyzer to determine whether the OBJ slot in the matrix clause unifies with the missing slot in the indirect question or with the indirect question as a whole.

Note that the internal syntax of indirect questions is almost identical to that of relative clauses. The distinction between relative clauses and indirect questions is only in terms of the role they play in the enclosing clause, that is, in terms of their external requirements only. An indirect question fills a participant role while a relative clause only modifies one such role. While the missing slot in a relative clause has to be compatible with the noun phrase being modified, indirect questions will have a missing slot which will have to be compatible with the question word. In relative clauses we share the missing slot with the noun phrase being modified while in indirect questions, the missing slots will be left unfilled.

We have already noted that nominalized clauses may also have missing participants. See for example,

134) The company wants the agent to sell the machines

135) The company wants to sell the machines

In 135) the SUBJ of the selling predication has been left unspecified with the intention that the readers can understand this as being shared with the SUBJ of the outer clause namely, 'the company'. Thus the matrix clause could be taken as 'The company wants something' and the embedded nonfinite clause as 'the company to sell the machines'. When a participant is shared by two or more clauses, we would like to place the participant in one of these shared roles and borrow the same in the remaining roles. The question of where to place the shared participant by default is therefore of relevance here. Consider the sentences

136) Krishna wanted me to join the army

137) Krishna told me to join the army

138) Krishna inspired me to join the army

We have said earlier that every clause must have a SUBJ and thus 'me' must certainly be a part of the embedded clause in all these example. Does this group have a role to play in the matrix clause also? Should we read the matrix clause in 136) as 'Krishna wanted something' or as 'Krishna wanted someone to do something'? Since the former interpretation also seems complete, one may be inclined to take that interpretation. If taken in the sense 'Krishna wanted that I should join the army', 'me' has no role to play in the matrix clause. On the other hand, 'Krishna told something' gives a sense of incompleteness - 'to whom did he tell?'. We must therefore take 'me' as the OBJ2 in the outer clause also. Do you inspire somebody or do you inspire somebody to do something?. We do not intend to provide appropriate answers to such questions here but we should realize that sharing rules must consider both of these possible alternatives - the SUBJ of the embedded infinitival clause may also, or may not, take on the OBJ2 role in the outer clause. Also, clauses can be nested more deeply and the sharing of participants can cut cross several clauses. Look at

139) Krishna wanted me to remind Radha to remember to try to solve the problem

Here 'me' is the SUBJ of 'remind' clause and possibly the OBJ2 of 'want' clause. 'Radha' is the OBJ2 of 'remind' and SUBJ of 'remember', 'try' and 'solve' clauses. With these issues in mind, we can write down the rule for sharing of roles in embedded infinitival clauses in English:

RULE : The missing SUBJ of an embedded infinitival clause in English will be shared with the OBJ2 of the outer clause if it is present, else it would be shared with the SUBJ of the outer clause, recursively.

If we work from whole to part, from the matrix clause into the embedded clauses, this rule can always be applied directly. The qualification 'recursively' ensures that this sharing rule will work even if infinitival clauses are embedded recursively over many levels.

There are other constructions where otherwise mandatory participants can be missing. One example is the passive construction. We have observed that the deep subject may be left unspecified in a passive sentence. In UCSG no attempt is made to 'transform' a passive construction into the supposedly equivalent active construction as many other grammar formalisms advocate. Knowing that a clause is in the passive voice - the verb group itself tell this, we simply permit the deep subject to be missing. Because of the interplay between dative shift and passivization, either OBJ or OBJ2 roles may be missing in the double object constructions:

140) Radha was given a book by Krishna
141) Radha was given a book

142) A book was given to Radha by Krishna
143) A book was given to Radha

There are other constructions where no participant is really missing or shared by more than one clause but some participants are displaced from their usual linear position. Clearly this happens only in positional languages - in relatively free word order languages linear position is not significant for F-Structure analysis. English questions are examples of this phenomena. The displaced participant invariably appears earlier than the place where it fits in. The simplest strategy for dealing with these displaced participants is therefore to simply 'hold' them temporarily and use them up later when required. There can be one or more intervening clauses between the place of appearance and the place of use of such participants and hence this is an example of what is called a long distance dependency. Long distance dependencies will form the subject matter of the next section. We will show that English questions in particular and displaced constituents in general can be handled using the same mechanism that we use for handling any kinds of inter-clause dependencies.

2.5.4 Long Distance Dependencies

Sharing and displacement of participants can span across clauses nested several levels deep. Consequently, the linear distance between the position where the displaced or shared slot appears overtly and the place where it is to be used can become arbitrarily large. This phenomenon has been termed long distance dependency. English questions and relative clauses are examples. Consider the declarative sentence and a question that can be formed from that

144) The boy read ‘Knowledge of Language’

145) Which book did the boy read?

The verb ‘read’ requires an object and ‘normally’, that is, in a declarative construct, the object appears after the verb in English. Hence we could say that the phrase ‘which book’ has been displaced from its original position on the right side of the verb to the sentence initial position. One good way of dealing with this situation is to ‘move’ the phrase back into its original position so that the sentence can then be analyzed without requiring any other change in the grammar. In simple sentences such as the above, there will be only one clause. The position where the sentence initial wh-phrase is to be moved can be determined the same way the position of the shared constituents in relative clauses is determined. The wh-phrase itself provides useful information - a ‘when’ question phrase gets into the TIME role and a ‘which’ phrase can get into either the SUBJ or the OBJ role and so on. The wh-phrase can only move into a role that is required but found to be missing. Of course all the other F constraints should also be satisfied.

Long distance dependencies occur because clauses can be inserted between the linear positions before and after movement. When there are several clauses in between, we need to be able to identify into which clause the wh-phrase moves before its role in that clause can be determined. UCSG offers an extremely simple and elegant solution to this problem. Consider the following questions and their declarative counterparts

146) Which book did Krishna think that the boy read?

147) Krishna thought that the boy read ‘Knowledge of Language’

148) Which book did Krishna think that Radha believes that the boy read?

149) Krishna thought that Radha believes that the boy read ‘Knowledge of Language’

150) Which book did Krishna think that Radha believes that Ram knows that the boy read?

151) Krishna thought that Radha believes that Ram knows that the boy read ‘Knowledge of Language’

Observe that the question phrase ‘which book’ plays the OBJ role in the ‘read’ clause, in all these questions. The functional structure of the ‘read’ clause is not affected in any way by inserting clauses in between. Also observe that all the intervening clauses are clauses that go into the OBJ slots of their respective outer clauses. The verbs in these clauses are all cognitive verbs. Pragmatically, what we ‘know’ or ‘think’ or ‘believe’ can be a whole predication and that is why we have clausal objects. Of course that clause can also involve a cognitive verb and hence can have another clause as its OBJ, leading to recursive nesting of clauses. The most important point to note is that as a rule this kind of nesting of clause occurs only with the OBJ role. No other role in the F-Structure exhibits this phenomenon. This very simple yet crucial observation has somehow been missed out in other grammar formalisms. Given this observation, the rule for ‘movement’ in English questions is extremely simple :

Rule : A displaced constituent either fits within the current clause or the current clause has a clausal OBJ and the displaced constituent fits in that clause, recursively.

The same rule holds for long distance movements in relative clauses. In the following examples the object-relativization rl ‘whom’ requires a missing OBJ role. In the first example OBJ of ‘saw’ is missing as expected. In the second example, the OBJ in the relative clause is another clause whose OBJ in turn is missing. The last example shows a two level transfer.

152) The girl wondered whom the nurse saw

153) The girl wondered whom the nurse claimed that the baby saw

154) The girl wondered whom Krishna believed that Radha claimed that the baby saw

Long distance dependencies are really a matter of hierarchical structure of clauses and not of linear order of constituents. All grammar formalisms based directly or indirectly on phrase structure rules for functional structure analysis will find it difficult to deal with long distance dependencies since phrase structure rules are inseparably tied up with linear order. Grammar formalisms have failed to recognize this simple fact and have endeavoured to 'solve the problem' of long distance dependencies using special mechanisms. There is really no problem to solve at all. Long distance dependencies have nothing to do with distance, long or short.

Two things are essential for elegantly handling the so called 'long distance dependencies'. Firstly we need to determine the hierarchical structure of clauses before we take up functional structure analysis so that we know which clauses are embedded within which other clauses. Secondly we need to analyze the functional structure starting the main clause and going down the hierarchy of clauses in a work-from-whole-to-part fashion so that expectations about missing constituents can be passed down the clause hierarchy. Also, displaced constituents, which invariably appear before they are needed, can be passed down the hierarchy of clauses so that they can simply be 'used up' where required. This is exactly how UCSG works.

Yes-no questions in English are same as declarative sentences except for the fact that there is an auxiliary shift. When an yes-no question is detected, we undo the auxiliary shift and parse the sentence exactly as we parse a declarative sentence. As we have shown above, we can easily find the matrix clause verb group in any sentence using the constraints on sequences of verb groups and sentinels as encoded in the H-Structure grammar rules. Once we find the matrix clause verb group, we can shift back the auxiliary to its correct position and get back the complete verb group easily. Wh-questions are simply yes-no questions with a wh-phrase removed and displaced to the sentence initial position. Therefore to parse wh-questions we simply call the same procedure which handles yes-no questions with an additional parameter that indicates the displaced wh-phrase and what kinds of roles it can fit in.

It is significant that no other grammar formalism advocates an independent hierarchical structure analysis. TAG however, handles long distance dependencies elegantly by 'adjoining' entire trees within other trees. TAG starts with the main clause and adjoins other clauses to obtain the required clause structure. Thus TAG is very similar to UCSG in the way it handles long distance dependencies. However, TAG operates on entire trees, including noun phrases, whereas to know the clause structure just the verb groups and sentinels are sufficient as we have shown. Thus UCSG is much simpler and far more efficient than TAG. LFG chose the inelegant direct linking in preference to the successive transfer of missing role information down the clause hierarchy. ATN used its more natural 'hold mechanism' to 'hold' a displaced constituent until it is needed in some clause. The held constituent could be used up in any role in any clause it fitted. The hold mechanism itself did not say anything about what kind of role this constituent can play or in which clause. In contrast UCSG imposes constraints on the clauses and roles in those clauses in which a 'held' constituent can be used.

2.5.5 Functional Structure Analysis

Functional structure analysis is the third and the last phase in UCSG. All potential groups will have been obtained in the linear analysis phase and all the clauses will have been identified and the clause structure obtained during the hierarchical analysis phase. Clause boundaries will also have been partially determined. What remains is the assignment of functional roles to the groups in each of the clauses. Word groups are treated as atomic units as far as such a role assignment is concerned.

In UCSG, functional structure analysis is done essentially one clause at a time. We limit our attention to the local domain of each clause to analyze its functional structure. It may be noted that no other grammar formalism posits an independent H-Structure component. In all these formalisms the search space includes the whole of the sentence.

For each clause, the verb group will be known. The subcategorization frame attached to the head of this verb group will tell us which participants in this predication are mandatory, optional and prohibited. For all the mandatory and optional participants, their syntactic type (noun group, indirect question, 'that-clause', etc.) will be indicated. Additionally, selectional restrictions in the form of semantic features may be given. The default surface case markers will also be given for the various roles. Default values may change depending upon the form of the verb group according to specified rules. The Paninians call such rules 'karaka chart transformations'. All these constraints provide a top-down or expectation driven flavour to functional structure analysis. The surface case marking information from the already identified word groups provides the data driven or bottom-up flavour. We thus bring to bear a combination of top-down and bottom-up strategies for pruning the search for possible functional role assignments. In the process, weaker constraints like grammatical agreement and semantic feature compatibility can also be verified.

In positional languages the linear position of the participant groups relative to the verb group indicates the functional roles they take whereas in relatively free word order languages the corresponding information comes from the morphological inflections and pre/postpositions. Apart from this, there is no difference between the two classes of languages. For analyzing the cross serially nested clauses in Dutch, all that we need to do is to allocate the noun groups in a left to right order instead of looking for proper nesting of verb groups and the corresponding noun groups.

Thus the F part of the grammar is a set of constraints and parsing is essentially a search for sets of role assignments that satisfy all the conditions simultaneously. Functional structure analysis can therefore be viewed as a constraint satisfaction problem. Various formal techniques exist for this. Bharati et al [4] have suggested, for example, integer programming, Assignment Problem (as in Operations Research) and bipartite graph matching as such possible general techniques for the role assignment problem.

We propose that the functional structure analysis start from the main clause. From hierarchical analysis, we would know the local domains for the various clauses. We limit our search to the local domains of the respective clauses. We start by attempting to assign the mandatory and optional roles to those constituents which lie within the specified local domain. If some of these constituents happen to be clauses themselves, we recursively go into those clauses and apply the same procedure until the entire sentence has been analyzed. If the constituents happen to be nominalized clauses we can analyze them in a similar way. However, the hierarchical structure analyzer may not provide clause boundary information for nominalized clauses. In any case, the local domain of the nominalized clause has to lie within the local domain of the enclosing clause. In the case of positional languages we may constrain the local domain further. For example, if we encounter a gerundial clause in a preverbal position in English, the local domain of the gerundial clause cannot go beyond the verb group of the enclosing clause. We pass down information about displaced, shared and missing participants down the hierarchy so that the functional structure analyzer has all the required information to analyze the current clause and clauses nested within it. The groups in the grey area may or may not get assigned. When we finish analyzing all the clauses in the sentence, the exact clause boundaries will have been obtained as a byproduct. We can verify that all the groups in the sentence have been allocated roles and no group takes more than one role. In the case of ambiguous sentences, the search can be continued until all possible parses are obtained.

We are thus working from whole to the part. We know the overall shape of a sentence, in terms of its clause structure, before we get into the details. Similarly we know something about a clause before we go into its internal details. We will know the verb group, one of the clause boundaries and the grey area on the other end of the clause. Working from whole to part is a strategy of traditional wisdom in many disciplines. Every civil engineer knows that this is the only technique which works in surveying. Every artist knows that he can't get the proportions right if he starts from one end proceeds linearly. The fact that human beings invariably process natural languages linearly from left to right notwithstanding, we see that working from whole to part has much to offer both for the grammar and for the parser. Working from whole to part means that at any given point of time we have a global view of what we are doing and yet our search space is constrained to a small local area around the focus. Information about displaced, shared and missing participants flows down the hierarchy of clauses. Thus hierarchical structure analysis has an extremely important role to play. Working from whole to part strategy is crucial for effectively dealing with displaced, shared and missing participant constructions and long distance dependencies.

Examples of Functional Structure Analysis

Let us see how our example sentence of chapter 1 repeated below is analyzed. From hierarchical structure analysis we will have obtained the local domains of all the clauses as indicated below.

155) The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night.

```
[(s,1,16,21),[(f_clause,3,4,4),[(vg,3,4,_)]],[(sub_clause,4,16,21),
 [(sb,4,5,_)],[(s,7,16,21),[(f_clause,7,16,21),[(rel_clause,7,9,13),
  [(rl,7,8,_)],[(s,8,9,13),[(f_clause,8,9,13),[(vg,8,9,_)]]]],
   [(vg,13,16,_)]]]]]
```

We start our analysis from the outermost s. The H-Structure says that we have a verb group lying between positions 3 and 4. We find the word 'believed' in this position. The L-Structure tells us that the root form of this verb is 'believe'. We look at the subcategorization frame for 'believe'. We notice that this verb requires a noun group as a SUBJ and an OBJ that can be a noun group or a preposition group with preposition 'in' or a 'that clause'. OBJ2 and SUBJ_QUAL are prohibited. We have only one noun group to the left of the verb group and hence 'the police' becomes the SUBJ. L-Structure says that the word 'that' can be a noun group by itself. If we attempt taking 'that' as a noun group we will have to account for all the remaining words in the sentence in terms of optional modifiers such as space and time. This not being possible, we will have to take the entire 'that clause' as the OBJ. From H-Structure we know that this clause starts at position 4 and ends somewhere between 16 and 21.

We now get into this nested clause. Here we find from H-Structure that the verb group is 'must-have-escaped'. From L-Structure we know that the root form of the head of this verb group is 'escape'. The verb 'escape' requires a noun group SUBJ and prohibits OBJ, OBJ2 as well as SUBJ_QUAL. Form H-Structure we see that there is a relative clause to the left of the verb group. The relative clause starts at 7 and ends somewhere between 9 and 13.

We in turn go into this relative clause and find its verb group to be 'wanted'. The verb 'want' requires a noun group SUBJ and an OBJ that can be either a noun group or a 'that clause' or an infinitival clause. The sentinel 'who' indicates that this is a case of subject-relativization. We find a missing subject between the sentinel and the verb group. Hence we simply borrow the noun group immediately to the left of the sentinel, namely, 'the dacoits' as the SUBJ. From L-Structure we know that there is an infinitive to the right and hence we get into the embedded infinitival clause. The verb group there requires an OBJ. Hence 'the bank' becomes the OBJ of 'to rob'. Since the infinitival clause is embedded in the relative clause, we also now know that the relative clause spans all the way up to the end of 'bank'. The infinitival clause lacks a SUBJ and the rule we have given earlier helps us to borrow it from the SUBJ of the outer clause, which we now know to be 'the dacoits'.

Since the relative clause extends all the way up to the verb group 'must-have-escaped' the SUBJ here must be the 'dacoits' which is in turn modified by the relative clause. What remains to be analyzed are the groups 'into the forest' and 'at night'. The preposition 'at' indicates either SPACE or TIME but the noun 'night' is marked in the lexicon with the TIME feature. Hence 'at night' is TIME. Similarly 'into the forest' becomes SPACE. Such semantic features are very useful for assigning roles to prepositional phrases. We often use the names of the functional roles themselves as the semantic features. Many prepositions indicate both time and space but there are only a limited number of time nouns. All the time nouns can be easily marked in the lexicon. Since the F-Structure roles are single valued, assigning specific roles greatly reduces the total number of possibilities.

The analysis of sentences in relatively free word order languages goes on exactly the same lines except that we use morphological information to assign roles. In our equivalent Telugu example we have analyzed the group 'doocaalanuko' equivalent to 'wanted to rob' as a single verb group.

The F-Structures of the entire sentences as they appear on the computer screen were given in chapter 1. Notice that we display the functional structure clause by clause and indicate the inter-clause dependencies through numbers. Shared constituents are shown blinking on the screen. The various roles are displayed in fixed locations on screen. This makes a good visual impact. One can quickly recognize the structure of a clause. One can clearly see linear, hierarchical and functional structure inherent in the sentence. We also retain information about the actual word order used in the input sentence and display for each clause the relevant portion of the sentence. Compare this with the more usual tree structures. The format in which the F-Structure will be stored in the output file is given below.

F-Structure 1 :

1 : The police believed 2

2 : that the dacoits 3 must have escaped into the forest at night

3 : who wanted 4

4 : to rob the bank

1 : subj : The police
1 : vg : believed
1 : obj : 2
    2 : cls_link : that
    2 : subj : the dacoits
    2 : subj_mod : 3
        3 : cls_link : who
        3 : subj : (the dacoits)
        3 : vg : wanted
        3 : obj : 4
            4 : subj : (the dacoits)
            4 : vg : to rob
            4 : obj : the bank
    2 : vg : must have escaped
    2 : space : into the forest
    2 : time : at night

2.5.6 Efficiency Considerations

Functional structure analysis is essentially a constraint satisfaction problem. There are a variety of constraints. The relative significance of specific types of constraints varies drastically from language to language. Thus it is difficult to incorporate the constraints right into the algorithms and then compute the computational complexity. On the other hand if we assume a generate-and-test kind of an algorithm that first generates all possible combinations which then get filtered out through the constraints, algorithms become extremely complex. This latter approach is unrealistic. In practice the constraints are very strong and if we apply the constraints at the earliest opportunity and prune the search, we can get algorithms which are fairly efficient in time. Thus obtaining precise yet realistic complexity measures is a very difficult task in NLP. There are different viewpoints ranging from one extreme to the other [28, 48, 51].

Let us first consider the case where a given sentence is analyzed for its F-Structure by the generate-and-test strategy. Let n be the length of the sentence (that is, the number of words in the sentence) and c be the number of clauses in the sentence. Let us assume, for the sake of simplicity, that each clause has exactly r functional roles. We assume that all the roles are mandatory. Thus there are a total of $(c * r)$ roles to be assigned. Roles are assigned to groups of words. Each role is assigned to exactly one group and each group in the sentence is assigned exactly one role. Let us ignore the possibility of shared roles. Let us assume that word groups can involve only contiguous words. Thus groups are nothing but substrings. There are $\frac{n*(n+1)}{2}$ possible substrings. Hence we can have $O(n^2)$ groups in the worst case. Groups may overlap - some words may belong to more than one group. Of all the groups we have to select $(c * r)$ groups such that no word in the sentence is left out and no word occurs more than once. We are thus obtaining partitions of length $(c*r)$ from the set of all possible groups in the sentence. There will be a total of $O(2^{n-1})$ partitions if all the $O(n^2)$ groups are formed. From binomial theorem we can see that there will be ${}^{n-1}C_r$ partitions of length $(n-r)$. We take one partition of length $(c*r)$ and assign $(c*r)$ roles to the groups. There are $(c*r)!$ ways of assigning $(c*r)$ roles to an equal number of groups. Each of these will have to be generated and tested against all the functional constraints. Will have to repeat the process for each of the partitions of the required size. Generate-and-test is a hopelessly inefficient technique.

There will be a great saving if we analyze the sentence clause by clause. Assume that we have a method of breaking a given sentence into c clauses and we can analyze these clauses completely independent of one another. Let us say each of the c clauses has n_c words in it. There will now be only $O(n_c^2)$ groups and $O(2^{n_c-1})$ partitions to consider in each clause. We will have to select partitions of size r and allocate the r groups in it to the r roles. This allocation will require only $(r)!$ cases to be generated and tested. We will have to analyze all the c clauses this way. Since $c * (r)!$ will be much smaller than $(c * r)!$ when $c > 1$, dividing a sentence into clauses and analyzing the clauses one by one is going to greatly reduce the complexity of functional structure analysis of complex sentences.

Even with this reduction, however, generate-and-test is extremely inefficient and unrealistic. Let us see if we can apply the functional constraints while generating itself. Let us look at the proposals made by the I.I.T. (Kanpur, India) group in their work on Paninian grammar [4]. They analyze the functional structure of the entire sentence at one go, not clause by clause. They make several assumptions. They assume that there are no ambiguities between nouns and verbs. There are no ambiguities in groups formation - there is only one partition of groups in the sentence. They further assume that each verb has only one subcategorization frame. The subcategorization frame specifies which roles are mandatory, which roles are optional. The roles not mentioned under either of these categories are presumed to be prohibited. For each role, the subcategorization frame further specifies the surface case marking information. Roles are to be assigned to the various noun groups such that all the mandatory roles of each verb group in the sentence are assigned and each noun group gets exactly one assignment. Optional roles may or may not get assigned. Thus the number of word groups in the sentence must be greater than or equal to the total number of mandatory roles of all the verb groups but less than or equal to the sum of all mandatory and optional roles in the whole sentence. Under these assumptions, they show that the role assignment problem can be modeled as a maximal matching problem in bipartite graphs or as an assignment problem. They quote [37, 1, 25, 30] and say that polynomial time algorithms exist.

The reality is however, more complex. Paninians accept that there are many words that can be both nouns and verbs. They say that we can still retain the polynomial nature of the solution and carry out functional structure analysis separately for each combination of noun-verb ambiguities. They say that as far as Indian languages are concerned, groups can be identified without any ambiguities. This assumption is clearly not valid in English and many other languages. Thus their approach cannot be taken to be universal.

More importantly, the Paninian grammar formalism advocates the analysis of the entire sentence in one go. In contrast, UCSG advocates that hierarchical analysis of clauses be determined first. Functional structure analysis can then benefit from the localizing effect of knowing the clauses and the clause boundaries. Clauses in a sentence are mostly independent of one another. Each clause has its own verb group. The verb group has its own expectations specified in the form of subcategorization frames. Each clause has its own set of noun groups that meet these demands.

Of course there are dependencies between clauses. All these dependencies are, however, related to the hierarchical structure of clauses. Two clauses can have any dependency between them if and only if one of them is nested inside the other, either directly or any number of levels deep. Therefore, if we determine the clause structure and carry out functional structure analysis clause by clause starting from the main clause and going recursively into the nested clauses, we can pass on all the dependency information down the hierarchy. This enables the functional structure analysis to be done completely in isolation of any other clause. To analyze any one clause only the groups within the boundaries of that clause need to be considered and only the functional roles in that particular clause need to be considered. In UCSG clause boundaries are only partially known, however. Groups in the grey area may belong to either of the adjacent clauses. All the groups must be assigned a role and no group can take more than one role except when dictated by syntax. But for this global constraint, functional structure analysis in UCSG is completely local to the respective clauses.

In UCSG we understand that there may be several ways of grouping words and we obtain all the potential word groups first. We have seen that all the potential groups in a sentence of length n can be obtained in $O(n)$ time. We analyze the clause structure using only verb groups and sentinels. Let the number of verb groups and sentinels in the sentence be n'. We have shown that hierarchical structure analysis can be done in $O(n'^3)$ time without making any assumptions about the depth of nesting or the number of clauses that can be embedded within a clause. We have got an efficient way of obtaining the clause structure by not considering noun groups and functional structure constraints. We do pay a price for this - we cannot determine the clause boundaries completely. The local domains of clauses may involve grey areas the words falling within which may have to considered in both of the bordering clauses. Also, as a consequence of lexical ambiguities in sentinels and verb groups, the hierarchical structure analyzer may produce more than one H-Structure some of which may have to be considered and then given up during functional structure analysis. Let us say h different H-Structures are produced and let ${n_c}'$ be the number of words per clause including the words in grey areas. Let us assume that functional structure analysis can be done in polynomial time with an unknown exponent x. The overall time complexity in UCSG can then be written as

$$O(n) + O(n'^3) + (h * c * O({n'_c}^x)$$

Since no algorithm with complexity less than $O(n^3)$ is known, as long as $x \geq 3$, $n' \leq n$, n'_c not much greater than n_c, and h is reasonably small, UCSG will be faster than all other algorithms. UCSG achieves this efficiency by analyzing the hierarchical structure of clauses first and then analyzing the functional structure clause by clause. The increase in efficiency is basically due to the fact that the c term is moved out of the exponentiated term.

We have found that n' is about $n/3$ on the average even in the presence of lexical ambiguities. Constraints on verb group-sentinel sequences are very strong and in most of the cases only a few H-Structures are produced. There can be several different valid ways of grouping words and we have to consider all the potential groups in a clause. However, the subcategorization frame, the surface case marking information on these potential groups and the information about displaced, shared or missing roles if any passed down from the outer clause form sufficiently strong constraints to prune the search effectively. Also, groups lying in the grey area do not increase the complexity significantly. Groups mandatorily required by one of the two adjacent clauses will get their role assignments straight away. Multiple possibilities arise only when some of these groups fill up optional roles in both the clauses. In some cases the sentence itself may be ambiguous. For example, an adverb group between two clauses may belong to either clause and we will have to consider all the alternatives in any case. The UCSG approach seems to be the best bet.

Chapter 3

The UCSG Parser

In this chapter we will give details of the implemented parsers for English and Telugu. The English parser was developed over the past few years concurrently with theory development. A sample Telugu parser was then developed in less than three days by adapting the already implemented English parser. Both the parsers handle a variety of simple as well as complex sentences. The English parser handles, among other things, active and passive constructs, dative shift, imperatives, yes-no and wh-questions, adverb embedding within verb groups, small clauses, infinitives without 'to', etc. The Telugu parser handles verb-less sentences and implicit subject constructions also. Further developmental work is going on for enhancing the coverage and robustness. As of now the UCSG parser for English is in a relatively more advanced state of development compared to the Telugu parser. These parsers are currently being used for other research projects in this department including research in spelling and grammatical error detection and correction, metaphor interpretation and machine translation.

Both the parsers have been developed in Arity Prolog version 6.1 to run on personal computers running the DOS operating system. Arity Prolog provides interpreter as well as compiler environments for software development. The interpreter along with its powerful debugger forms an ideal tool for development and testing. Optimized compiled versions can then be built for efficiency.

Arity Prolog is an extendible language. Thus the lexicon, morphological analysis rules, transition networks forming the L part of the grammar, the H grammar rules and subcategorization frames can be either read in from data files or made internal to Prolog. In the English parser, the main lexicon and subcategorization frames are internal to Prolog. Custom files can also be used for including additional data as well as for overriding built-in data where required. The H rules are read in from a data file and the rules are analyzed to obtain look ahead and other pieces of information useful for making the parser more efficient.

The parsing system is implemented as four modules named UCSP, L, F and F2. The UCSP module includes the main program and the H part of the parser - the most language invariant part of syntax. L and F modules deal respectively with linear and functional structure analysis. F2 includes the user interface routines related to the functional structure analysis, which is again invariant across languages.

The main program opens up windows, issues title and start up messages and reads in the required data files. It then opens up a strategies screen in which strategies for user interaction can be specified. The L, H and F modules are executed in that order. The main program maintains timing information and it produces a histogram showing the average time taken to parse sentences of given length.

The linear structure analyzer breaks sentences into words, expands contractions, notes down possible proper names, looks up the virtual lexicon, builds all potential word groups and initializes the active chart for use by the hierarchical structure analyzer, all in a single scan of the sentence. In the current implementation of the English parser, morphological rules for plural forms for nouns, past tense and present participial forms for verbs and degrees of comparison for adjectives have been included. For unknown words, the system can obtain the required information from the user and continue with parsing. The words in the lexicon are hashed and a software cache is employed for speed.

Hierarchical structure analysis is carried out using a variant of the *active chart parser*. Active chart parsing combines the best of top-down and bottom-up parsing strategies. Partial results are stored in the chart and reused so that nothing is done more than once. The chart also stores the minimal spans of all the constituents. This helps to obtain the clause boundaries easily.

In the current implementation H grammar rules are given in the form of the 12 CFG rules that we saw in the previous chapter. The correct non-nested structures are obtained from the structures produced by these recursive rules during parsing itself. Whenever a rule of the type A $\rightarrow$ (a A) is applied during parsing, the non-nested structure corresponding to the rule A $\rightarrow$ (a a) is also simply added. This applies at all instances of the A-rule application and the correct structure can be obtained even when there are several levels of nesting of clauses. It should be made clear that we are not building complete parse trees and then carrying out 'transformations' on them. Tree transformations are in general costly. All we need is to build the appropriate structure as and when a rule is applied. This is a simple check done locally - at the place and at the time where a rule is applied. There is no need for searching in trees.

The functional structure analyzer starts from the main clause and works into the nested clauses. Information about displaced, shared and missing participants is passed as a parameter down the hierarchy of clauses. In each clause the verb group of that clause is picked up from the hierarchical structure and the root form of the head of the verb group is obtained from the linear structure. The subcategorization frame is then retrieved. For newly added verbs, subcategorization frames can be added interactively. From linear structure we would know the potential word groups that fall within the local domain of the clause. The hierarchical structure provides information about sub_clauses that may fill some of these roles and about rel_clauses that may modify roles filled by noun groups. All possible role assignments are made consistent with these constraints. Agreement checks are made only after possible F-Structures are obtained. Shared roles are shown blinking on the screen. Roles which are required by the subcategorization frames but permitted to be missing by syntax are shown empty.

Where there are more than one possible F-Structures, all the F-Structures are produced. In the current version there are no facilities for prioritizing or ranking the possible F-Structures. The natural ability of Prolog to backtrack and seek alternative solutions is exploited to obtain all possible F-Structures. In cases where the hierarchical structure analyzer has produced more than one H-Structure, functional structure analysis is carried out for each of these H-Structures. The current implementation handles simple and complex sentences but it does not handle sentences with coordinate conjunctions or comparatives.

UCSP-Telugu :

Adapting the already implemented English grammar and parser for Telugu was very simple and fast. The entire process took less than three days. This speaks for our claims on the universal nature of UCSG. In particular, the H part of the grammar required only the few parametric changes as indicated in the previous chapter. The parsing algorithm for the H part required practically no change. The L part of the grammar that deals with grouping of words had to be changed to suit the structure of Telugu word groups. Telugu is rich in morphological inflections. Whatever English achieves by grouping together individual words, Telugu obtains the same effect by simply inflecting a single word in many situations. For example, verb groups are most of the times single words in Telugu. The grammar of word groups, the L grammar, is thus relatively simple in Telugu. The L part of the parser required no change. The sample parser bypassed the morphological component by including complete word forms in the lexicon directly. Telugu morphological analyzers already developed here [47, 45] are now being integrated with the Telugu parser. The F part needed only one significant change. The bottom-up constraints for role assignment in Telugu come from morphological inflections rather than from linear position of word groups. The F part of the parser was modified accordingly. All the other parts of the complete system including the user interfaces remained unchanged. The GIST script processor card was used for viewing in Telugu script.

Chapter 4

Epilogue

We end this thesis with concluding remarks and directions for further research work. We also mention the directions in which UCSG parsers need to be developed further. Our primary aim was to develop a computationally viable grammar formalism which enables both positional languages and relatively free word order languages to be parsed efficiently. The theoretical model of UCSG that we presented in chapter two, supported by practical implementations of English and Telugu parsers, shows that we have met our primary objectives.

By dividing the grammar as well as the parser into L, H and F modules, we have come to notice that positional languages, relatively free word order languages and even languages like Dutch that exhibit cross-serial dependencies differ from one another mainly in the F component. The division into three modules has helped us to identify and localize the differences. We can exploit the commonness in the L and H modules for both theoretical and practical advantage.

Western grammar formalisms have taken linear position constraints all too seriously. The fact that subjects and objects are indicated by linear position in some languages does not justify the degree of importance that is attached to linear position in these grammar formalisms. This obsession for linear position has prevented these theories from taking a straight look and discerning the universal features common to positional languages and relatively free word order languages.

Also, the modularization into L, H and F in UCSG has given us an edge in terms of simplicity of grammar and efficiency of parsing. In the past, when it was shown or believed that grammars of a particular type were inadequate to deal with the syntax of human languages, this was simply taken to imply that more powerful grammars were called for. We instead ask which aspects of the syntax of our languages are handled elegantly by which type of grammar. Our L module has a linear time complexity and the H module has a worst case time complexity of the cube of the number of verb groups and sentinels in the sentence. We have further argued that the H module can also be linearized since both the depth and branching factor of the H-Structure trees are bounded and hence the 'language of H-Structure' is a finite language. We have also shown that UCSG is more efficient on the whole compared to any other grammar formalism that does incorporate an independent H module and cannot benefit from the localization of functional structure analysis.

Many of the arguments provided in the past for the insufficiency of a particular type of grammar were based on infrequent and obscure examples rather than on the run-of-the-mill phenomena. For example, the cross-serial nesting in the 'respectively' construct in English was taken to imply that the whole of English is not context free. While the major debate in the past was whether context free grammars are sufficient or not, we have shown that there are significant portions in the syntax of natural languages which can be handled by an even more simpler grammar - the finite state grammar. We have shown that the linear structure inherent in the word groups can be obtained using finite state machine power and in linear time. It is significant that no other grammar formalism achieves linear time parsing in any of their components. Also, word groups are appropriate and meaningful units of structural description and hence it is improper to deal with syntax directly in terms of words. Words must be grouped into appropriate groups. This grouping also makes all subsequent steps in syntactic analysis simpler and more efficient since we will then be dealing with the significantly shorter sequences of groups rather than with the words in the sentence.

We have shown that analysis of the hierarchical structure of clauses in a sentence require both linear position and recursive nesting constraints. Context free grammars are therefore ideally suited for this purpose. H-Structures are trees. Context free grammars are therefore essential as well as just sufficient. We have shown that hierarchical structure analysis can be done based only on the verb groups and sentinels to the total exclusion of all noun groups. Only a few rules are required. Thus hierarchical structure analysis becomes very efficient. We have shown that we can obtain all of the possible H-Structures and even obtain clause boundary information. Although we cannot obtain all the clause boundaries precisely, we can obtain the grey areas and the local domains for each of the clauses involved. Also, since the branching factor in the H-Structure is limited by the number of functional roles in a clause that can take clausal participants and the depth of the tree is limited by psycholinguistic resource constraints in the human brain, H-Structures are finite structures in practice. All finite languages are regular. Hence hierarchical structure analysis can also be linearized.

More importantly, we have shown that the H-Structure is the most language invariant component. Hence the name Universal Clause Structure Grammar. The H grammars of different languages show only parametric variations. H-Structure should really form the core of any syntactic theory. An independent H level analyzed before and without applying the F level constraints is the highlight of the UCSG grammar formalism. No other grammar formalism incorporates an independent H level.

Functional structures of individual clauses in a sentence are mostly independent of one another. Knowing the H-Structure enables us to localize the functional structure analysis to the local domains of the respective clauses. The limited dependencies that exist between the functional structures of different clauses are all fully captured by the H-Structure. All the inter-clause dependency information flows down the H-Structure. Thus by carrying out functional structure analysis starting from the root of the H-Structure tree and proceeding towards the leaves, we can pass on the relevant dependency information also, making functional structure analysis almost completely local to individual clauses. The only remaining global dependencies are the constraints that all the words in the input sentence must be analyzed and no word be assigned more than one role. We have shown that this localizing effect reduces the complexity and increases the efficiency of functional structure analysis. We have also shown that parsing becomes efficient on the whole.

As a byproduct of this working from whole to part strategy, we get rid of all problems of long distance dependencies. Inter-clause dependencies have something to do with hierarchical structure but they have nothing whatever to do with linear distance, long or short. By working down the H-Structure tree and passing on relevant information down the tree, we make the analysis of each clause completely local. All dependencies effectively become intra-clause dependencies.

We have argued that it is better to handle syntactic variants of constructs by noticing the nature and location of the variation and 'patching up' rather than by positing separate rules for all possible constructs. We have indicated how reduced constructions obtained by dropping certain optional constituents can also be handled uniformly by the same grammar that handles the full forms of these constructions.

We have thus achieved our dual objectives of universality and efficiency by dividing the problem of syntactic analysis of natural language sentences into three independent modules. We also get many other benefits. The division into three separate modules makes grammar simpler and easier to write. The L part is quite easy to write and the H part is largely common across languages.

Our parsers depend mostly on syntactic information. Semantic features are used sparingly. They are used if and only when they help us to select the correct F-Structures to the exclusion of the others. Semantic information is used only when the gains are much more than the cost. Even though we depend mostly on syntactic information only, our parsers produce only a few parses. This is a result of a number of strategies. Firstly, we use all the available information effectively. We encode in the subcategorization frames the syntactic types of constituents that can appear in the various functional roles. Also, while we look at the word 'at' as simply a preposition for L analysis, for F analysis we make use of the more detailed information that the word 'at' indicates time or space. We look at 'who' simply as sentinel marking relative clauses at H level but at F level we observe that 'who' indicates subject relativization. Secondly, we employ a small set of more general functional roles so that ambiguities in role assignment get reduced. We make use of the functional structure constraint that no role can take more than one participant to rule out many possibilities. Thirdly, we do not analyze the groups in detail for their internal hierarchical or functional structure. We have argued that such an analysis is not possible in general unless large amounts of semantic and pragmatic knowledge is brought to bear. As an extreme example, the noun phrase in example 51 in section 2.3.2 above that is quoted to have 429 different parses, will have a single interpretation in UCSG if that noun phrase happens to be the subject of a sentence. Lastly, we do not intend to obtain the correct word senses in all situations. We feel that this is again beyond syntax. Thus ‘Mary had a little lamb’ will have only one F-Structure in UCSG despite the fact that the word ‘had’ has a number of different senses. F-Structures are links between syntax and semantics. We have chosen the draw the line of demarcation closer to syntax in order to achieve our major objectives. It is better not to bite more than what we can really chew.

Lexical ambiguities abound in all languages and no syntactic framework can afford to ignore them completely while claiming to be practicable. UCSG handles all lexical ambiguities at L level. Structural ambiguities arising out of prepositional phrase attachment are not directly handled but the one role - one participant constraint at the F level often eliminates some possibilities. Subordinate clause attachment is also a semantic and pragmatic level problem and we will just be content with producing all the possible parses.

The structural descriptions that UCSG produces clearly depict linear, hierarchical and functional structure. Unlike in the more usual trees, words, phrases, clauses and sentences have their own independent levels in the structure. The parsers display functional structure clause by clause and shows the functional roles in fixed locations on the screen for visual impact and more effective communication.

We have not made any cognitive claims in this thesis. We do not assert that human beings work the same way. We do believe, however, that human beings use simple and efficient techniques. It is human to solve complex problems by dividing it into simpler subproblems. It is a natural human ability to work from whole to part. We would not be surprised much if others take up the issues of cognitive plausibility of UCSG and find that there is something human in it.

Another worthwhile task is to analyze and exploit the inherent parallelism in UCSG. For example, each of the potential H-Structures can be analyzed for its F-Structure in parallel.

The robustness of linguistic based systems can be enhanced by incorporating a statistical component. Statistical analysis helps to concentrate on the most likely possibilities and to exclude cases with very low probabilities.

Currently, UCSG does not handle coordinate conjunction, comparatives, direct speech and sentence fragments. Efforts are on to enhance the coverage and robustness of the implementations. The parsers are currently being applied to research in spelling and grammatical error detection and correction, metaphor interpretation and machine translation between English and Indian languages.

Bibliography

[1] R K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[2] M Bates. The Theory and Practice of Augmented Transition Networks. In L Bolc, editor, *Natural Language Communication with Computers*. Springer, 1978.

[3] Akshar Bharati, Medhavi Bhatia, Vineet Chaitanya, and Rajeev Sangal. Paninian Grammar Applied to English. *To appear in: Computer Science and Informatics*, Special issue on Natural Language Processing and Machine Learning, 1995.

[4] Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India, 1995.

[5] Rajesh Bhat. Paninian Theory for English. B.Tech. thesis, I.I.T.(Kanpur), 1994.

[6] L Birnbaum. *Integrated Processing in Planning and Understanding*. PhD thesis, Dept. of Computer Science, Yale University, 1986. Research Report No. 489.

[7] Joan Bresnan, Ronald M Kaplan, Stanley Peters, and Annie Zaenen. Cross-Serial Dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635, 1982.

[8] Noam Chomsky. Three Models for the Description of Language. *IRE Trans. on Information Theory*, 2(3):113–124, 1956.

[9] Noam Chomsky. On Certain Formal Properties of Grammars. *Information and Control*, 2(2):137–167, 1959.

[10] Noam Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, Massachusetts, 1965.

[11] Noam Chomsky. *Reflections on Language*. Pantheon, New York, 1975.

[12] Noam Chomsky. *Lectures on Government and Binding*. Foris, Dordrecht, 1981.

[13] Noam Chomsky. *Knowledge of Language: Its Nature, Origin and Use*. Praeger, New York, 1986.

[14] Longacre R E. *The Grammar of Discourse*. Plenum Press, 1983.

[15] Jay Earley. An Efficient Context-Free Parsing Algorithm. *cacm*, 13(2):94–102, 1970.

[16] Charles J Fillmore. The Case for Case. In Emmon Bach and Robert T Harms, editors, *Universals in Linguistic Theory*. Holt, Rinehart and Winston Inc, New York, 1968.

[17] Charles J Fillmore. The Case for Case Reopened. In Peter Cole and Jerrold M Sadock, editors, *Syntax and Semantics*, volume 8: Grammatical Relations. Academic Press, 1977.

[18] Gerald Gazdar. Unbounded Dependencies and Coordinate Structure. *Linguistic Inquiry*, 12(2):155–184, 1981.

[19] Gerald Gazdar, Ewan Klein, Geoffrey K Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Harward University Press, Cambridge, Massachusetts, 1985.

[20] Gerald Gazdar and Geoffrey K Pullum. Computationally Relevant Properties of Natural Languages and their Grammars. In William Marsh Walter J Savitch, Emmon Bach and Gila Safran-Naveh, editors, *The Formal Complexity of Natural Language*. D. Reidel Publishing Co., Dordrecht, 1987.

[21] Amrey Gethin. *Antilinguistics - A Critical Assessment of Modern Linguistic Theory and Practice*. Intellect, Oxford England, 1990.

[22] J Greenberg. Language Universals. In T A Sebeok, editor, *Current Trends in Linguistics - III Theoretical Foundations*, pages 61–112. The Hague Mouton, 1966.

[23] Morris Halle, Joan Bresnan, and George A Miller. *Linguistic Theory and Psychological Reality.* MIT Press, Cambridge, Massachusetts, 1978.

[24] James Higginbotham. English is not a Context Free Language. *Linguistic Inquiry*, 15(2):225–234, 1984.

[25] J E Hopcraft and R M Karp. A $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs. *SIAM Journal of Computation*, 2:225–231, 1973.

[26] Geoffrey Horrocks. *Generative Grammar.* Longman, London, 1987.

[27] Aravind K Joshi. Tree Adjoining Grammar: How much Context-Sensitivity is Required to Provide reasonable Structural Descriptions. In L Karttunen D Dowty and A Zwicky, editors, *Natural Language Parsing.* Cambridge University Press, Cambridge, UK, 1985.

[28] G Edward Barton Jr, Robert C Berwick, and Eric Sven Ristad. *Computational Complexity and Natural Language.* The MIT Press, Cambridge, Massachusetts, 1987.

[29] Ronald M Kaplan and Joan Bresnan. Lexical-Functional Grammar: A Formal System of Grammatical Representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Massachusetts, 1982.

[30] H W Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[31] Mitchell P Marcus. *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, Massachusetts, 1985.

[32] William Marsh and Barbara H Partee. How Non-Context Free is Variable Binding? In William Marsh Walter J Savitch, Emmon Bach and Gila Safran-Naveh, editors, *The Formal Complexity of Natural Language*. D. Reidel Publishing Co., Dordrecht, 1987.

[33] K. Narayana Murthy. Parsing Telugu in the UCSG Formalism. In *To appear in: Proceedings of the Indian Congress on Knowledge and Language*, 1996.

[34] K Narayana Murthy and A Sivasankara Reddy. INQUIRER: A Language Independent View of Natural Language Understanding. In R M K Sinha, editor, *Computer Processing of Asian Languages*, pages 174–181. Tata McGraw-Hill, 1992.

[35] K Narayana Murthy and A Sivasankara Reddy. INQUIRER: A Case System Based on Questions. In E Balagurusamy and B Sushila, editors, *Innovative Applications in Computing*. Tata McGraw-Hill, 1993.

[36] K Narayana Murthy and A Sivasankara Reddy. Universal Clause Structure Grammar. *Communicated to: Computer Science and Informatics*, Special issue on Natural Language Processing and Machine Learning, 1995.

[37] Christos H Papadimitrou and K Steiglitz. *Combinatorial Optimization*. Prentice-Hall, New Jersey, 1982.

[38] Geoffrey K Pullum and Gerald Gazdar. Natural Languages and Context-Free Languages. In William Marsh Walter J Savitch, Emmon Bach and Gila Safran-Naveh, editors, *The Formal Complexity of Natural Language*. D. Reidel Publishing Co., Dordrecht, 1987.

[39] G Uma Maheshwara Rao. Morphological Formations of Telugu Verb. unpublished manuscript, University of Hyderabad, 1995.

[40] James Rogers and K Vijay-Shankar. Reasoning with Descriptions of Trees. In *ACL92: Proc. of the Annual Meeting of the Assoc. of Computational Linguistics*, 1992.

[41] James Rogers and K Vijay-Shankar. Obtaining Trees from Their Descriptions: An Application to Tree-Adjoining Grammars. *Computational Intelligence Journal*, 1993.

[42] S W Russel. Semantic Categories of Nominals for Conceptual Dependence Analysis of Natural Languages. Technical Report Stanford A.I. Memo number 172, Computer Science Dept. Stanford University, 1972.

[43] Roger C Schank. *Conceptual Information Processing*. North-Holland, 1975.

[44] Peter Sells. *Lectures on Contemporary Syntactic Theories*. Center for the Study of Language and Information, SRI International, California, 1985.

[45] C Ravi Shankar. *Database Access in Telugu*. PhD thesis, Department of Computer and Information Sciences, University of Hyderabad, 1993.

[46] Stuart M Shieber. Evidence against the Context-Freeness of Natural Language. *Linguistics and Philosophy*, 8:333–343, 1985.

[47] V Sreeram. Design and Development of a Morphological Analyzer for Telugu Language. Master's thesis, Department of Computer and Information Sciences, University of Hyderabad, 1995.

[48] Masaru Tomita. *Efficient Parsing for Natural Language: A Fast Algoithm for Practical Systems*. Kluwer Academic Publishers, Boston, Masachusetts, 1985.

[49] L G Valiant. General Context-Free Recognition in Less Than Cubic Time. *Journal of Computer and System Sciences*, 10(2):308–315, 1975.

[50] K Vijay-Shankar. Using Descriptions of Trees in a Tree Adjoining Grammar. *Computational Linguistics*, December 1992.

[51] William Marsh Walter J Savitch, Emmon Bach and Gila Safran-Naveh, editors. *Computationally Relevant Properties of Natural Languages and their Grammars*. D. Reidel Publishing Co., Dordrecht, 1987.

[52] Terry Winograd. *Language as a Cognitive Process: Vol1: Syntax*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

[53] William A Woods. Transition Network Grammars for Natural Language Analysis. *cacm*, 13(10), October 1970.

[54] D H Younger. Recognition and Parsing of Context-Free Languages in time n^3. *Information and Control*, 10(2):189–208, 1967.