Universal Clause Structure Grammar

K. Narayana Murthy* & A.Sivasankara Reddy*

This paper is about a new grammatical formalism which we call Universal Clause Structure Grammar (UCSG). In this paper we present the essence of UCSG with emphasis on efficient parsing of both positional languages like English and relatively free word order languages such as the Indian languages. Grammars can be viewed as sets of constraints on the structure of sentences. There are constraints on the linear position of constituents, hierarchical nesting of constituents one inside the other and functional dependencies between different constituents in a sentence. In UCSG 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. UCSG proposes three levels of representation called L-Structure, H-Structure and F-Structure along with the corresponding components of the grammar to map from one level to the other. In UCSG we divide and conquer. The division of labour into these three separate modules, especially the introduction of an independent H-Structure is the highlight of UCSG. UCSG claims that strong constraints exist on the sequences of verbs and certain clause boundary markers, and that the hierarchical structure of clauses in a sentence should be analyzed by exploiting these constraints. UCSG not only shows that the clause structure of a sentence can be determined without and before analyzing the predicate-argument structure of individual clauses, it also demonstrates the following three major advantages of doing so. Firstly, parsing in UCSG is very efficient.

Secondly it provides a uniform method of dealing with configurational languages like English and relatively free word order languages like Indian languages and even languages like Dutch that show cross serial dependencies. Finally, it disparages the notion of long distance dependencies. Brief comparisons with other grammar formalisms are also given.

1. Introduction:

The grammar of a language is a formal specification of the valid structures in that language. The aim of a syntactic analyzer, also called a (syntactic) parser, is to apply the grammar to produce a structural description of a given sentence. Producing a structural description of the given sentence helps in understanding the meaning of the sentence and facilitates Natural Language Processing (NLP) applications such as machine translation. The tasks for a syntactic analyzer include identifying the various constituents in the sentence, determining their linear and hierarchical relationships and assigning functional roles to these constituents.

Grammars and the processes of parsing and generating are closely tied up. A grammar formalism is a meta language that specifies the nature of grammars, parsers and structural descriptions. Different grammar formalisms can be compared in terms of the efficiency of parsing or generation in those grammar formalisms. More powerful grammars provide higher flexibility in specifying the grammatical structure of languages but they also require more computational resources for parsing or generation and in this sense less efficient. The natural goal in NLP is, therefore, to look for the least powerful and hence the most efficient grammar that is sufficient to deal with the structure of natural languages.

There is a notion of a Universal Grammar - a grammar that aims to capture the underlying commonalities among all human languages and attempts to explain human language behaviour in terms of such a grammar [3,4,7]. While Chomsky views Universal Grammar as an in-born biological endowment, others like Greenberg view universality

Keywords: Grammar Formalism, Universal Grammar, Parsing

* *Department of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046 INDIA*

purely in terms of underlying commonalities between different languages despite the idiosyncrasies of individual languages. We take this latter view - we make no cognitive claims. From a practical point of view, the notion of a Universal Grammar is attractive because it enables one to build universal parsers and generators that need only relatively minor changes for adapting them into individual languages. This would require much less effort compared to building separate parsers and generators for individual languages. Common parsers and generators also imply that the structural descriptions that a parser produces or a generator accepts will be uniform across languages and hence automatic translation from one language to another would be greatly facilitated. In particular, here we are interested in grammar formalisms that can efficiently handle both positional languages like English and relatively free word order languages like the Indian languages.

Aim and Scope:

Our aim here is to develop a computationally viable grammar formalism that leads to efficient parsing for both positional languages and relatively free word order languages. We develop a new grammar formalism which we call Universal Clause Structure Grammar (UCSG) and show how efficient parsers can be built within this grammar formalism.

In the next Section we give the necessary background for UCSG. We discuss basic concepts including the notions of phrases, clauses and linear, hierarchical and functional structure. We will then see how other grammar formalisms have dealt with these kinds of structure and what kinds of problems they face. In the light of this discussion, we present the UCSG grammar system in some detail. We will see how UCSG deals with relatively free word order languages and with long distance dependencies. We will also show that efficient parsers can be built for UCSG. We conclude with a detailed example.

2. Background:

We use a (simple) sentence to describe a predication- an action or the state of something. Typically, we use verbs to specify the action or the state itself and nouns to specify the various participants in the predication. Thus the English sentence

1) *I bought a computer*

can be ascribed 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. Predicate-argument structure is the heart of structural descriptions.

2.1 Clauses and Hierarchical Structure:

To express more complex ideas we may combine several predications into a single sentence. Complex sentences may involve several inter-related clauses. Clauses may be nested one inside the other in several ways and several levels deep. Clauses can function as subjects and complements of other clauses. Also, the various participants in a predication may be further modified by relative clauses. Sentences in natural languages therefore exhibit a hierarchical structure of clauses. This notion of hierarchical structure is crucial for UCSG and UCSG differs from other grammar formalisms mainly in terms of how it captures this hierarchical structure in sentences.

We look at a clause as a linguistic unit that corresponds to one predication. A clause includes one verb group that specifies the predicate, all the arguments of this predicate as also all the non-arguments (modifiers of place, time, etc) that go along with the predicate. Consider

2) *The computer which I bought from the famous company is defective.*

Here the main or the matrix clause is 'The computer is defective'. One of the arguments in this clause, namely, 'the computer' is further being modified by the relative clause 'which I bought from the famous company'.

Here 'The computer' is an argument at the matrix clause level but 'the famous company' is not. In order to understand the structure of the sentence properly, it is essential for a parser to be able to identify the clauses, to determine the clause boundaries and to find out inter-clausal relationships.

2.2 Functional Structure:

Functional structure of a sentence relates to the predicate-argument structure of the various clauses in a sentence. It indicates the assignment of functional roles to the various participants in each of the predications being made in the sentence. The primary claim of a functional structure is that there are only a small number of universal roles into which we can analyze any sentence from any human language - functional roles themselves are language independent. Small number of roles implies a high level of abstraction. Only one assignment can be made to a role, each role that the subcategorization frame demands must be filled and every constituent in the given sentence plays one functional role except

for specific cases where sharing of roles is permitted according to well specified rules of grammar. These are all the well known tenets of the so called 'case grammars'. Different case systems use different sets of case roles. Panini's karaka roles are perhaps the oldest. Within western linguistics, the notion of surface and deep cases were first clearly laid out by Fillmore [5,6] and since then roles such as agent and theme have become standard in linguistic theories.

2.3 Word Groups and Linear Structure:

Arguments need not be single nouns, they can be groups of words that together have the force of a noun. It is therefore useful to introduce the concept of a phrase - a group of words that plays, as a whole, one role in the sentence. In the sentence

3) *The new chip had become the standard*

the phrase 'the new chip' is a single argument. This phrase as a whole has a role in the predicate-argument structure of the sentence and its parts cannot play any role on their own. Every phrase has a 'head' and a phrase takes the name of its head. Every noun phrase is headed by a noun or in fact, any other word that has the force of, or that can stand for, a noun, such as a pronoun. Parts of a phrase may or may not have independent meaning. The word 'the' in 'the new chip', for example, may not have any independent meaning although it may contribute to the overall meaning of the phrase. A phrase as a whole can usually be ascribed some meaning, can be given out as a fragmental reply to a question and usually translates as a unit to other languages.

Likewise, the group of words 'had become' has a meaning as a whole – the meaning of 'had become' is not merely a composition of the meanings of the words 'had' and 'become', and can be translated as a unit into some other language. Consequently, we should call 'had become' as a verb phrase – it is a phrase headed by a verb. By the same token 'keeps coming' and 'used to come' are single verb phrases since there is no sense of 'keeping' or 'using' in these phrases.

Unfortunately, it has become traditional to divide a sentence into a 'subject' and a 'predicate' and refer to the entire 'predicate' of the sentence (including the verb group and the 'objects') as the verb phrase (vp). To avoid confusion, we will use the term 'verb group', rather than the term 'verb phrase'. For the sake of uniformity, we will also refer to other phrases like noun phrases as noun groups and so on.

Phrase structure rules capture repetition through recursive rules. Such rules and the corresponding tree structures give us an impression that we are capturing hierarchical relations also. This is in fact quite misleading. Noun phrases with multiple modifiers do exhibit hierarchical structure in terms of what modifies what but determining the correct modifier-modified hierarchy is a difficult problem that relies heavily on semantics as can be seen from examples like 'glass epoxy printed circuit board manufacturing unit'. It is therefore reasonable to expect syntax to only list all the modifiers preserving linear order, without pretending to have analyzed the hierarchical structure. Recursion in these phrase structure rules is to be interpreted only as an implementation of iteration - permitting more than one modifier. Word groups involve constraints on linear precedence, optionality and repetition but no hierarchical nesting. See [13] for more on this issue. On the other hand, the linear order of phrases within a clause may or may not be significant. English imposes much stricter ordering of the arguments in a predication than do the relatively free word order languages like Telugu.

The complete structural description of a sentence should therefore depict the predicate-argument structure of each of the clauses in the sentence along with the inter-clause relationships. It should also show the linear position of the constituents in the sentence wherever significant. We will now briefly discuss how other grammar formalisms have tried to capture linear, hierarchical and functional structure in sentences. We shall discuss some of the problems with these approaches. This discussion will help us to appreciate the merits of the UCSG approach to syntactic analysis.

3. Capturing Linear and Hierarchical structure :

Most grammar formalisms handle linear and hierarchical structure together using a component equivalent to Context Free Grammars (CFG). Functional structure is usually handled by some sort of augmenting or enhancing the CFG component. As we shall see later, UCSG differs from this approach.

Most grammar formalisms, including Augmented Transition Networks (ATN) [17], Definite Clause Grammar (DCG) [15], Lexical Functional Grammar (LFG) [10], Tree Adjoining Grammar (TAG) [9], the older Transformational Grammar and the more recent linguistic models like Government and Binding (GB) [3,4], incorporate a component that

deals with the linear and hierarchical relationships in sentences. ATNs are obtained by augmenting Recursive Transition Networks (RTN) which are equivalent to Context Free Grammars. LFG uses phrase structure rules to capture the linear and hierarchical relationships which are depicted in its C-Structure. TAG uses initial and auxiliary trees which depict linear as well as hierarchical relationships. Linguistic models like GB incorporate a base component that uses phrase structure rules to capture linear and hierarchical relationships. In all these grammar formalisms the component that deals with linear and hierarchical relationships uses grammars of the CFG power. Context Free Grammars are well understood in computer science and efficient parsing techniques exist for parsing with CFGs. However, inappropriate use of CFGs can lead to both theoretical and practical problems. We now turn to a detailed discussion of the merits and demerits of CFGs so that we can clearly see where we should use them and where we should not.

1. *Dependencies among constituents in a sentence:*

Languages exhibit certain dependencies among the various constituents in a sentence including grammatical agreement and selectional restrictions. Phrase structure rules are not good at capturing dependencies among constituents. CFGs relate to words through grammatical categories. The category of a word is, and should be, based purely on its own intrinsic properties. On the other hand, a grammatical relation between two independent words is an intrinsic property of neither of these words. One can handle these dependencies by dividing the categories - for example nouns and verbs can be divided into singular nouns, plural nouns, singular verbs and plural verbs to deal with number agreement between a noun phrase and the verb. But this is unintuitive and computationally highly inefficient. CFGs cannot deal effectively with any kind of functional dependency.

2. *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 phrases in a sentence without significantly altering the functional structure of the sentence. Indian languages are examples of this. Phrase structure rules, have no way of 'ignoring' this unwanted linear order constraints, since the right hand side of a phrase structure rule not only specifies the constituents but also their linear order. We are again forced to multiply the rules writing one rule for each possible order of the constituent phrases. This is a case where CFGs impose linear structure where there is none. This makes all the western grammar formalisms unsuitable for parsing relatively free word order languages. On the other hand, while the Paninian Grammar (PG) [1] has been especially developed for Indian languages, its applicability to positional languages has not yet been fully established. Moreover, PG suffers from many of the same problems that western grammar formalisms suffer from and it has been argued that Paninian grammar, even if applied to positional languages like English, will lead to very inefficient parsing and in fact it is less efficient than possible even for Indian languages [13].

3. *Long Distance Dependencies and Movement:*

Consider the sentence and the question that follows:

4) *That the new computer which I had purchased from the famous company had a hardware bug has been troubling me*

5) *Whom has the fact that the new computer had a hardware bug been troubling?*

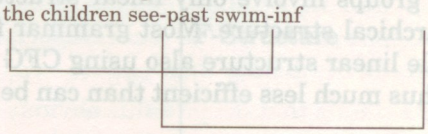
In interpreting this question, we may say that the 'wh' phrase 'whom' has to be 'moved' to its proper position, which is to the right of 'troubling'. Since we can interpose any number of clauses such as the relative clause in 4), we call this kind of dependency as a 'long distance dependency'. Since CFGs cannot deal effectively with any kind of dependencies among constituents, handling long distance dependencies is also problematic. All the problems of long distance dependencies are simply a consequence of not making a clear distinction between clauses and phrases and trying to handle both using a monolithic set of phrase structure rules.

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 a nested constituent must end before the outer constituent closes. There are languages such as Dutch, however, where we come across an infinite set of grammatically correct sentences with cross serial nesting. Let us look at some examples from Dutch, taken from [2].

6)

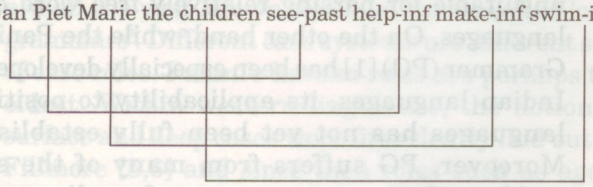
... dat Jan de kinderen zag zwemmen
that Jan the children see-past swim-inf



'...that Jan saw the children swim'

7)

...dat Jan Piet Marie de kinderen zag helpen laten zwemmen
that Jan Piet Marie the children see-past help-inf make-inf swim-inf



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

There are restrictions on cross serial nestings in Dutch [2], the most important of which is that only a verb that is subcategorized for both a NP and an infinitival complement can be inserted in this cross serial fashion. Bresnan et al [2] have shown that it is possible to get a CFG that sufficient to handle Dutch in terms of its weak generative capacity but they have also shown that Dutch is not strongly context free. The LFG solution to this problem is also given in [2]. TAG also handles cross serial dependencies but is more powerful and hence less efficient than CFGs [9].

4. *Unbounded Branching:*

CFGs capture repetition through recursion. There are situations where what we want is mere repetition and not recursion. For example, items in conjunction should all be at the same level. A sequence of 'a's can only be captured using recursive rules such as A -> a A or A -> A a leading to imposition of non-existent hierarchical structure. Otherwise we will have to use an infinite number of rules such as A -> a, A -> a a and so on.

In summary, we must use CFGs if and only if both linear and hierarchical structure are significant and there is no functional dependency. If there is no hierarchical structure, we may not even need the power of CFGs and if there is no linear structures, CFGs are no good. Most grammar formalisms make no distinction between phrases and clauses and employ a monolithic set of CFG rules or equivalent to capture both linear and hierarchical structure. This confusion between linear and hierarchical structure makes the western grammar formalisms unsuitable for handling relatively free word order languages and also leads to problems of long distance dependencies. As we have seen above, phrases or word groups involve only linear structure and no hierarchical structure. Most grammar formalisms handle linear structure also using CFG power and are thus much less efficient than can be.

4. Capturing Functional Structure :

Determination of functional structure involves assignment of roles to the various constituents in the sentence. This assignment is governed by the subcategorization frames of the verbs including the selectional restrictions. Verbs have their expectations, also called 'Akanksha' in the traditional Indian grammars. The constituents that fill the required roles must possess the necessary qualities. This is called 'Yogyata'. Additional information for role assignment comes from the prepositions or the post positional markers, morphological inflections and the linear order of constituents in the sentence.

Various clauses in a sentence are inter-related and clauses are not completely independent of one another. There can be, for example, shared constituents. Nevertheless, functional structure is essentially local to the individual clauses. Each clause has its own verb group, each verb has its own expectations and only the noun groups that are part of the particular clause can take on the various roles. This is simply as consequence of the fact that each clause corresponds to one predication. However, most grammar formalisms attempt to analyze the functional structure of the sentence as a whole. This makes these grammar formalisms very inefficient.

ATN builds a functional structure as a side effect of the 'actions' associated with the arcs. LFG builds a functional structure by unifying and solving the functional description equations that originate from the functional specifications associated with the right hand side symbols of the phrase structure rules and from the lexical entries. DCG is similar - it uses unification of feature structures. TAG claims that the functional structure can be simply read off from the derivation trees. Case grammars and the Paninian grammars view the task of producing a functional description as an assignment problem - assigning functional roles to various constituents subject to constraints specified above. In all these grammar formalisms, functional structure of the entire sentence is analyzed at one go. All these grammar formalisms fail to recognize the fundamental fact that functional structure is essentially a clause level phenomenon, not a sentence level phenomenon.

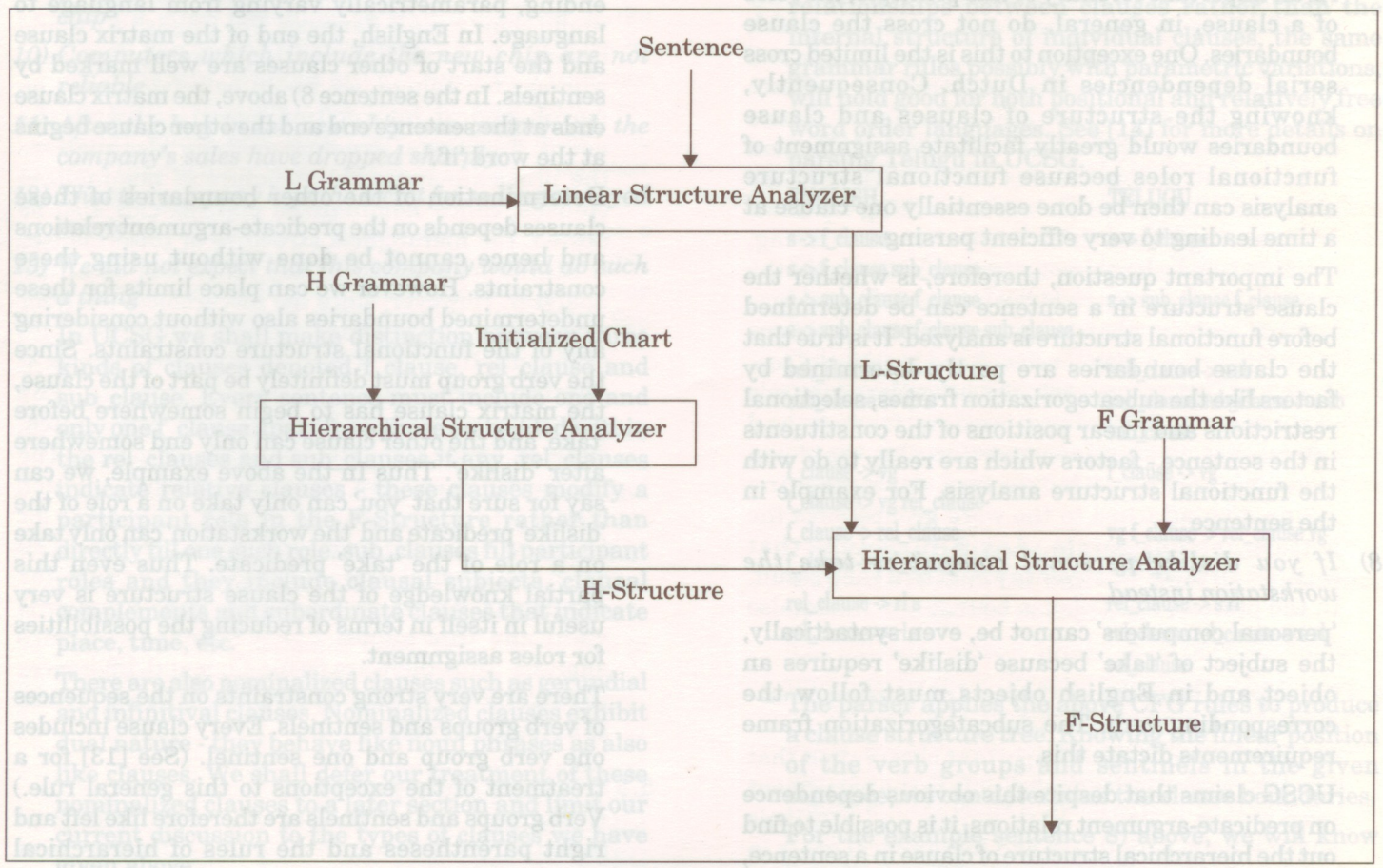
In summary, all these grammar formalisms are much less efficient than possible and none of them is equally well suited for handling both positional and relatively free word order languages. Much more efficient and uniform parsing is possible if the task of syntactic analysis is appropriately divided and

the right kind of grammar is employed to handle each of the modules. With these things in mind, we can now look at the UCSG grammar formalism in detail.

5. The Universal Clause Structure Grammar:

In UCSG we divide and conquer. By appropriately dividing the task of syntactic analysis into subtasks, we get to see where exactly relatively free word order languages differ from positional languages and what is really common between the two classes. We can also apply the least powerful kind of grammar for each subtask leading to highly efficient parsing on the whole. We find that the three primary kinds of structure inherent in human languages namely, linear, hierarchical and functional structure, lend themselves naturally for analysis by three independent modules. UCSG postulates three levels of syntactic analysis - linear analysis, hierarchical analysis and functional analysis. Correspondingly, there are three levels of syntactic representation termed L-Structure, H-Structure and F-Structure. While other grammar formalisms also tackle linear, hierarchical and functional structure in some way or the other, the division of labour into three separate modules, especially the introduction of an independent H level is the highlight of the UCSG grammar formalism.

UCSG recognizes that word groups act as atomic units of structure at both the hierarchical and functional structure level. Hence word groups are obtained first during the linear analysis phase. Then the hierarchical structure of clauses is determined. UCSG shows that hierarchical analysis of clauses in a sentence can be done without and before applying the functional structure constraints such as agreement, subcategorization frame and selectional restrictions. This approach makes it possible to efficiently and uniformly parse both positional and relatively free word order languages. Having obtained the hierarchical structure of clauses in the given sentence, functional structure analysis is done clause by clause making the parsing mechanism highly efficient on the whole. The block diagram below depicts the overall structure of UCSG. We will now discuss the details of these modules with emphasis on the hierarchical structure level, the most crucial aspect of UCSG.



5.1 The L-Structure:

The linear structure analyzer takes a sentence as input and produces its L-Structure. The main task of the linear structure analyzer is to identify all potential word groups in the given sentence. The linear structure analyzer obtains words, looks up the lexicon and carries out morphological analysis where required. It then identifies all potential verb groups, noun groups, prepositional groups, adjective groups, adverb groups etc. We can obtain all potential word groups in a sentence in a single left to right scan of the sentence and in linear time. Hence the name linear structure. See [13] for more details.

5.2 The H-Structure:

A general principle that is valid across all human languages is that in a sentence the arguments of a predicate must appear close to that predicate, only the notion of 'close' varies slightly from language to language. In English there are restrictions on the relative positions of the various arguments - the subject comes before the verb, the objects follow the verb, etc. In Telugu, the arguments can come in any order but all of them would come before the verb. As an important corollary of this principle, the roles of a clause, in general, do not cross the clause boundaries. One exception to this is the limited cross serial dependencies in Dutch. Consequently, knowing the structure of clauses and clause boundaries would greatly facilitate assignment of functional roles because functional structure analysis can then be done essentially one clause at a time leading to very efficient parsing.

The important question, therefore, is whether the clause structure in a sentence can be determined before functional structure is analyzed. It is true that the clause boundaries are partly determined by factors like the subcategorization frames, selectional restrictions and linear positions of the constituents in the sentence - factors which are really to do with the functional structure analysis. For example in the sentence

8) *If you dislike personal computers take the workstation instead*

'personal computers' cannot be, even syntactically, the subject of 'take' because 'dislike' requires an object and in English objects must follow the corresponding verb. The subcategorization frame requirements dictate this.

UCSG claims that despite this obvious dependence on predicate-argument relations, it is possible to find out the hierarchical structure of clause in a sentence, albeit partially, before and without considering any of the arguments or modifiers. UCSG shows that the hierarchical structure of the clauses can be efficiently determined by looking at only a few constituents in the given sentence. It also show that for each clause one of its boundaries can be fully determined and the other boundary can be bounded between two limiting positions. Having partially analyzed the hierarchical structure of the clauses, we can localize our search to the individual clauses for the assignment of functional roles to the arguments and modifiers.

By definition, every clause has one verb group in it and this must definitely be part of that clause. How do we determine which other constituents belong to this clause? There is an important observation that leads to the solution of this problem: One of the boundaries of every clause in a sentence is overtly marked by certain kinds of words or morphemes called sentinels. In English, for example, relative clauses begin with relative words like 'who', 'which' and 'that'. We will consider the beginning and the end of a complete sentence also as sentinels since they also mark clause boundaries. Sentinels may mark either the clause beginning or the clause ending, parametrically varying from language to language. In English, the end of the matrix clause and the start of other clauses are well marked by sentinels. In the sentence 8) above, the matrix clause ends at the sentence end and the other clause begins at the word 'if'.

Determination of the other boundaries of these clauses depends on the predicate-argument relations and hence cannot be done without using these constraints. However we can place limits for these undetermined boundaries also without considering any of the functional structure constraints. Since the verb group must definitely be part of the clause, the matrix clause has to begin somewhere before 'take' and the other clause can only end somewhere after 'dislike'. Thus In the above example, we can say for sure that 'you' can only take on a role of the 'dislike' predicate and 'the workstation' can only take on a role of the 'take' predicate. Thus even this partial knowledge of the clause structure is very useful in itself in terms of reducing the possibilities for roles assignment.

There are very strong constraints on the sequences of verb groups and sentinels. Every clause includes one verb group and one sentinel. (See [13] for a treatment of the exceptions to this general rule.) Verb groups and sentinels are therefore like left and right parentheses and the rules of hierarchical

structure enforce proper nesting of these parentheses. These are very strong constraints. For example, we can determine the matrix clause verb group by simply matching these parentheses. In English, for example, if you set a counter to zero, increment it every time you hit a sentinel and decrement whenever you see a verb group, you will be at the matrix clause verb group the first time the count becomes minus one. In the above example, the count becomes 1 at 'if', zero at 'dislike' and -1 at 'take' which is the matrix clause verb group. Handling auxiliary shift in English questions, for example, is thus very simple [13].

Let us consider clause structure in English in some detail. The simplest case is that of a sentence which has a single verb group in it. Here there is only one clause and that is also the complete sentence (example 9). Some of the arguments or even the modifiers of space, time, etc. may be further qualified by clauses, called relative clauses (example 10). There may be subordinate clauses that indicate place, time, etc (example 11). Clauses may be subjects or complements of a verb group in another clause (examples 12 and 13).

9) *The famous company sells computers with the new chip*

10) *Computers which include the new chip are not reliable*

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

12) *That the company kept the secret for so long annoyed everyone*

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

In UCSG we shall make distinction between three kinds of clauses denoted f_clause, rel_clause and sub_clause. Every sentence must include one and only one f_clause apart from those embedded within the rel_clauses and sub_clauses if any. rel_clauses indicate relative clauses - these clauses modify a participant role in the F-Structure rather than directly fill one such role. sub_clauses fill participant roles and they include clausal subjects, clausal complements and subordinate clauses that indicate place, time, etc.

There are also nominalized clauses such as gerundial and infinitival clauses. Nominalized clauses exhibit dual nature - they behave like noun phrases as also like clauses. We shall defer our treatment of these nominalized clauses to a later section and limit our current discussion to the types of clauses we have given above.

In English, the matrix clause has no explicit sentinel except possibly for the end of the sentence itself. The beginning of relative clauses is marked by the relative word sentinel 'rl' and the beginning of sub-clauses is marked by a subordinating conjunction sentinel 'sb'.

Sub_clauses standing for clausal subjects, clausal complements and modifiers of space, time, etc. can come before as well as after 'f_clause' in an English sentence. Within each 'f_clause', there can be relative clauses modifying the various arguments and non-arguments. There may be several sub_clauses and several rel_clauses nested recursively. We can capture these structures using the following CFG rules. These twelve CFG rules form the complete set of rules required for analyzing the hierarchical structure of English sentences - they are not mere samples. In fact rules of this kind apply to all languages with parametric variations. The number and nature of rules will be essentially same in all languages and hence the parser will also work with uniform efficiency. To illustrate this the rules for Telugu, a relatively free word order are also given. Since freedom of word order is a clause internal phenomenon and we are here dealing with the inter-relationships between clauses rather than the internal structure of individual clauses, the same grammar rules possibly with parametric variations, will hold good for both positional and relatively free word order languages. See [14] for more details on parsing Telugu in UCSG.

ENGLISH	TELUGU
s -> f_clause	s -> f_clause
s -> f_clause sub_clause	
s -> sub_clause f_clause	s -> sub_clause f_clause
s -> sub_clause f_clause sub_clause	
sub_clause -> sb s	sub_clause -> s sb
sub_clause -> sb s	sub_clause sub_clause -> s sb sub_clause
f_clause -> vg	f_clause -> vg
f_clause -> vg rel_clause	
f_clause -> rel_clause	vg f_clause -> rel_clause vg
f_clause -> rel_clause	vg rel_clause
rel_clause -> rl s	rel_clause -> s rl
rel_clause -> rl s	rel_clause rel_clause -> s rl rel_clause

The parser applies the above CFG rules to produce a clause structure tree. Knowing the linear position of the verb groups and sentinels in the given sentence, we can determine the clause boundaries.

For the example sentence 8) above, we will know

that the sub_clause begins at 'if' and ends somewhere after 'dislike' and before 'take'. The matrix clause begins where the sub_clause ends and it ends at the end of the sentence. Exact clause boundaries would get determined during functional structure analysis.

Since both the recursive nesting and linear precedence constraints embodied in these rules are significant, the power of CFGs is both necessary and just sufficient to deal with the hierarchical structure of sentences. CFGs are ideally suited for hierarchical structure analysis. They are more powerful than required for linear structure analysis and they are unsuitable for functional structure analysis.

5.3 The F-Structure:

In this last and final phase, functional roles are assigned to the various participants in each of the clauses in the input sentence. There have been a variety of case systems proposed by various researchers [1,5,6,8,11]. In UCSG we have used a system of functional roles developed keeping the question answering paradigm in mind [12]. Since the clause structure would have already been analyzed, the functional structure analyzer can work clause by clause. Since lower level clauses play specific roles in the higher level clauses in which they are nested, starting with the matrix clause and working down the hierarchy would greatly facilitate role assignment. Also, since the inter-clause dependencies including the sharing and displacement of constituents are all related to the hierarchical structure of clauses, working top-down and passing down information about missing, displaced and shared constituents will make the functional structure analysis completely a clause internal problem, thereby getting rid of all problems of long distance dependencies. A combination of top-down and bottom-up strategies is employed for role assignment. Subcategorization frames and selectional restrictions provide the top-down constraints while the surface case marking information attached to the word groups form the bottom-up constraints. Any F-Structure that satisfies all these constraints would be a valid result. In the process, exact clause boundaries would also get determined.

5.4 Relatively Free Word Order Languages and Cross Serial Dependencies:

In UCSG we make this important observation that freedom in word order applies only to the participant roles within a clause. Verb groups and sentinels follow strictly both linear and hierarchical order in all languages. Hence we can still use the same kind of CFG rules and determine the hierarchical structure exactly as we do for positional languages like English. See [14] as an example of UCSG being applied to Telugu, a relatively free word order language of south India.

We have also noted above that only nominalized clauses can be cross serially nested in Dutch. Nominalized clauses exhibit dual nature. They are noun phrases and clauses at the same time. In the UCSG view, recognizing noun phrases is the task of the linear analyzer and assigning functional roles is the task of the functional structure analyzer. The hierarchical analyzer in UCSG views these nominalized clauses as noun phrases and thus simply ignores them. So we do not need to make any changes to the hierarchical analyzer.

UCSG understands the powers and limitations of phrase structure rules, divides the problem into sub problems such that we get a separate component - the H-Structure, for which phrase structure rules of the CFG type are both essential and just sufficient. Thus the hierarchical analyzer in UCSG works uniformly and efficiently in all languages.

The F-Structure analyzer for different languages would show some differences. For English we would be using positional information for role assignment whereas for Telugu, we would be relying more on morphological information associated with the nouns. Functional structure analyzer also analyzes the predicate-argument structure of nominalized clauses. Since nominalized clauses can also be nested, the hierarchical structure of these clauses is also to be handled by the functional structure analyzer. However, since the hierarchical analyzer would have already analyzed the nested structure of clauses, the task is localized and the analysis of nominalized clauses is much simpler. The nominalized clauses in English are properly nested, while in Dutch they are cross serially nested. The functional structure analyzer employs a stack discipline for English and a queue discipline for Dutch for assigning roles to the noun phrases. In all cases the general and unifying principle that the arguments of a clause must come close together is valid and this guides the functional structure analysis. Thus there are no fundamental differences in functional structure analysis of different classes of human languages.

5.5 Efficiency Considerations:

It has been shown in [13] that identification of potential word groups involves only the aspects of

optionality, repetition and linear precedence. There is no hierarchical structure within word groups at the syntactic level. Hence the power of finite state machines is sufficient - we do not need the power of CFGs. Hence linear structure analysis can be done in linear time - the fastest ever possible. It is significant that no other grammar formalism has any component that is claimed to have linear time complexity.

We have shown that context free grammar is sufficient to capture the hierarchical structure of clauses. We have only a small number of rules. The length of the input string will also be much shorter than the length of the complete sentence since only verb groups and sentinels enter into hierarchical structure analysis. Typically, the length of this string is only about a third of the total sentence length despite lexical ambiguities. Thus, hierarchical analysis in UCSG will be highly efficient. Hierarchical structure analysis can be done with a worst case time complexity of the cube of the number of verb groups and sentinels in the sentence. In the current implementation, an active chart parser is used. It may be noted that ATN, DCG, LFG, TAG are all more powerful on the whole than CFGs.

Finally, since functional structure analyzer works clause by clause F-Structure analysis will also be very efficient. If there are c clauses in a sentence and each clause has n words on the average, UCSG parser would deal with the c clauses one by one, each time dealing with n words at a time, in contrast to other grammar formalisms where all the (n X c) words would have to be assigned roles in the c clauses all taken together. Thus parsing in UCSG is much more efficient than in other grammar formalisms like ATN, DCG, LFG, TAG and PG. See [13] for more details and quantitative analyses.

5.6 Long Distance Dependencies:

In UCSG, we note that the 'wh' phrase in an English question is only a noun phrase which is irrelevant for hierarchical structure analysis. Hence linear and hierarchical analysis go through exactly as for assertive sentences except for the auxiliary shift which can be easily handled since the matrix clause verb group is known. Once the clause structure is determined, the sentence initial 'wh' phrase has to be allotted a role either in the matrix clause or transitively in one of its embedded clauses. This is not difficult since we already know the clause structure. We combine information from the lexical items in the 'wh phrase' and the filled and unfilled roles in the matrix clause and in clauses embedded in it and assign the correct role. Thus the UCSG grammar does not require any special mechanisms and the so called long distance dependencies are handled in a simple and elegant way.

ATN uses the 'hold' mechanism and LFG uses special bounded domination meta variables with correspondence between the up and down meta variables. TAG handles long distance dependencies elegantly. In TAG since whole trees can be inserted by the adjoining operation, we get the same effect as in UCSG. In both UCSG and TAG, there are really no long distance dependencies - all dependencies are local to a clause and inserting any number of additional clauses in between does not change this property. TAG, however, uses trees and directly manipulates entire trees which include verbs, sentinels as well as all the noun phrases. Thus TAG grammar is more complex - more complex than need be. TAG is a mildly context sensitive grammar, whereas UCSG uses only CFG power for hierarchical structure analysis and then carries out functional structure analysis clause by clause.

5.7 An example: Annotated extracts from a transcript:

Given Sentence: The police believed that the dacoits who wanted to rob the bank must have escaped into the forest at night (Length: 20 words)

L-Structure analyzer: Identifies all potential word groups and passes to H-Structure analyzer a string of verb groups and sentinels only: 'believed that who wanted must-have-escaped' (Length: 5)

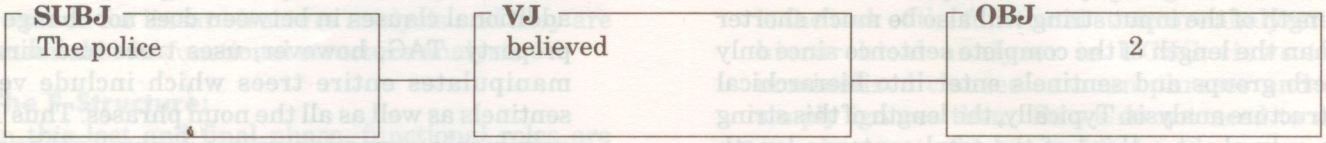
H-Structure analyzer: Using only the above five word groups and the twelve CFG rules, the H-Structure Analyzer produces all possible clause hierarchies one of which is shown below. Here curly braces indicate sub-clauses and square brackets indicate rel_clauses. There is one left bracket marking the exact starting position of the clause and two right brackets marking the limiting positions for the clause end point. Numbers associated with the brackets help to match corresponding left and right brackets.

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}

The following screen dumps show the final analysis produced by the F-Structure analyzer. The numbers at the top indicate the serial number of sentence being parsed, the parse number and the clause number. Clauses are numbered with the matrix clause always being numbered one. In clauses 3 and 4 the subject 'the dacoits' is borrowed from the outer clauses and is shown blinking on the screen.

UCSP-English

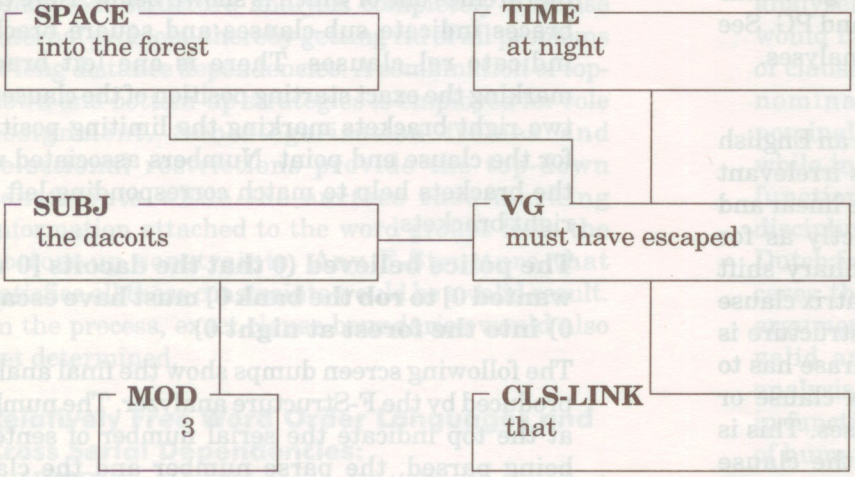
1 : 1 : 2 : The police belived 2



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

UCSP-English

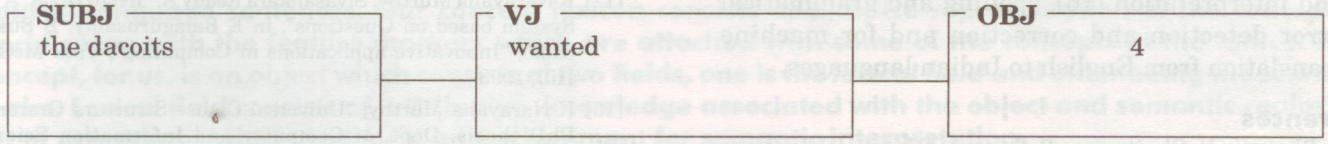
1 : 1 : 2 : that the dacoits 3 must have escaped the forest at night



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

UCSP-English

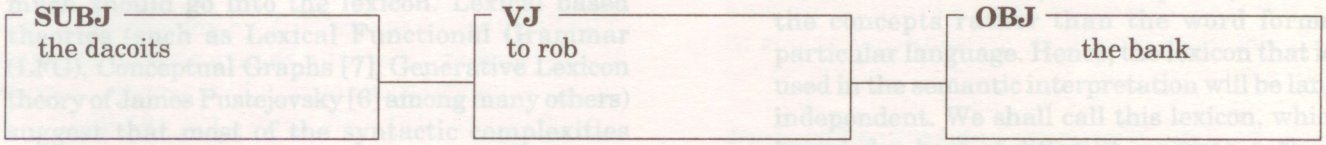
1 : 1 : 3 : Who wanted 4



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



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

6. Conclusions:

In this paper we have described the essence of the UCSG grammar formalism. We have discussed how the UCSG view can lead to highly efficient parsing of both positional and relatively free word order languages and without any of the problems of long distance dependencies. We have also indicated how cross-serial dependencies can be handled. We have shown that parsing in the UCSG system will be much more efficient than in other grammar formalisms. See [13] for more details. UCSG parsers are currently being used for metaphor recognition and interpretation [16], spelling and grammatical error detection and correction and for machine translation from English to Indian languages.

References

[1] A Bharati, V Chaitanya, R Sangal, "Natural Language Processing: A Paninian Perspective", Prentice-Hall of India, 1994

[2] J Bresnan, R M Kaplan, S Peters, A Zaenen, "Cross-serial Dependencies in Dutch", Linguistic Inquiry, vol 13, no 4, Fall 1982 pp 613-635

[3] N Chomsky, "Lectures on Government and Binding", Foris, Dordrecht, 1981

[4] N Chomsky, "Knowledge of Language: Its Nature, Origin and Use", Praeger, New York, 1986

[5] C J Fillmore, "The Case for Case", in E Bach, R T Harms (Eds), "Universals in Linguistic Theory", Holt, Rinehart and Winston Inc, 1968

[6] C J Fillmore, "The Case for Case Reopened", in P Cole, J M Sadock (Eds),"Syntax and Semantics", vol 8, Academic Press, 1977

[7] J Greenberg, "Language Universals" in T A Sebeok, (Ed), "Current Trends in Linguistics - III Theoretical Foundations", pp 61-112, The Hague Mouton, 1966

[8] J S Gruber, "Studies in Lexical Relations", Doctoral dissertation, M.I.T. 1965

[9] A K Joshi, "Tree Adjoining Grammars: How much context sensitivity is required to provide reasonable structural descriptions?", in D R Dowty, L Karttunen, A M Zwicky (Eds), "Natural Language Parsing", Cambridge University Press, 1985

[10] R M Kaplan, J Bresnan, "Lexical-Functional Grammar: A Formal System of Grammatical Representation" in Bresnan (Ed), "The Mental Representation of Grammatical Relations", M.I.T. Press, 1982

[11] R E Longacre, "A Grammar of Discourse", Plenum Press, 1983

[12] K Narayana Murthy, Sivasankara Reddy A, "INQUIRER: A Case System based on Questions", in E Balagurusamy, B Sushila, (Eds), "Innovative applications in Computing", Tata McGraw-Hill, 1993

[13] K Narayana Murthy, "Universal Clause Structure Grammar", PhD thesis, Dept. of Computer and Information Sciences, University of Hyderabad, 1995

[14] K Narayana Murthy, "Parsing Telugu in the UCSG Formalism", Proc. of Indian Congress on Knowledge and Language, vol II, pp 1-16, 1996

[15] F C N Pereira, D H D Warren, "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, vol 13, pp 231-278, 1980

[16] Vasudev Varma, "Tat-Tvam: A Metaphor Interpretation Model", PhD thesis, Dept. of Computer and Information Sciences, University of Hyderabad, 1996

[17] W A Woods, "An Experimental Parsing System for Transition Network Grammars", in R Rustin (Ed), "Natural Language Processing", Algorithmics Press, New York, 1973