Parsing Telugu in the UCSG formalism

K. Narayana Murthy
Department of Computer and Information Sciences,
University of Hyderabad
email:knmcs@uohyd.ernet.in

Abstract

This paper is about syntactic analysis of natural language sentences. We address the issues of the nature and extent of knowledge that is relevant for syntactic analysis. In particular, We show how the UCSG grammar formalism helps us to answer questions about the relative significance of different aspects of knowledge for the purpose of syntactic analysis. We take Telugu, a relatively free word order language, as an example for illustrating syntactic analysis within UCSG.

The UCSG grammar formalism enables simple and elegant grammars to be written and efficient parsers developed for both relatively free word order languages and positional languages. The crux of UCSG lies in the way it divides the problem. In UCSG we divide both the grammar and parser into three modules corresponding to the three primary aspects of structure inherent in human languages, namely linear (L), hierarchical (H) and functional (F) structures. This division makes the grammar simpler, more uniform across languages, and easier to write. The parser also becomes efficient since each module can employ the least powerful parsing mechanisms essential for the main task in that module. The division of grammar as well as the parser into these modules also helps us to clearly see which aspects of knowledge are really essential for handling the different aspects of structure.[1]

1 Introduction

Natural Language Processing (NLP) deals with computational models of human language processing. NLP views language as a means of communication [7]. There are two things that are essential for effective communication through language. Firstly,

[1]1.The research reported in this paper was supported by the Department of Science and Technology under grant no. SR/OY/E-10/94
2. The author wishes to acknowledge Dr. G Uma Maheshwara Rao and P S Vivekanand for Telugu language related expertise.

both the speaker and the listener must be active cognitive processors. More importantly, there must be a significant overlap in the knowledge they possess. Knowledge required for effective communication through language includes

- Knowledge of the language
- Knowledge of the situation or context
- Knowledge of the world in general

Communication breaks down if this overlap criterion is not satisfied. In the least, the listener must know the particular language being used. Also, if the necessary background knowledge is not available, either the listener does not understand at all or misunderstands what is being said. Hence what exactly is the knowledge that is essential for effective communication is an extremely important question in NLP.

NLP requires the modelling of various levels of processing such as lexical, morphological, syntactic, semantic and pragmatic levels. We focus here on the syntactic analysis of natural language sentences. A syntactic analyzer accepts one sentence and produces a structural description of the input sentence. The grammar of a language is a specification of valid structures in that language. A parser is a procedure that applies the constraints of the grammar to obtain the appropriate structural descriptions. A grammar formalism is a meta language - the language that specifies the structure of the grammar and the parser.

Syntactic theories usually deal with one sentence at a time and hence tend to ignore the contextual knowledge required for complete understanding. General knowledge of the world may be required to various extents to pick out appropriate structures. The knowledge of the language itself is of course crucial for syntactic analysis. The emphasis on different aspects of knowledge is to some extent theory dependent. In this paper we discuss the application of UCSG grammar formalism for parsing Telugu. The UCSG grammar formalism was developed with a view to facilitate efficient parsing of both positional languages like English and relatively free word order languages such as Indian languages. UCSG also facilitates the development of simple and computationally viable grammars for different languages. We will give the essence of UCSG in the next section before taking up syntactic analysis of Telugu in detail. See [2] for full details of UCSG.

2 The UCSG Grammar Formalism

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.

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 will call this aspect of structural description which depicts the functions or roles played by the various constituents as functional structure or F-Structure for brevity. Both mandatory and optional roles are depicted.

Predications manifest themselves as clauses in the surface representation of a sentence. Clauses are therefore considered to be appropriate units of structural descriptions in UCSG. 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. Also, relative clauses can be used to modify nouns. 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. The H-Structure of a sentence depicts the various clauses in a sentence, their hierarchical relationships as well as the clause boundaries.

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. Secondly, even words which have independent meaning often lose or change their intrinsic meaning when they combine with other words in a sentence. Most importantly, structural units that fill the functional roles in the F-Structure are in general groups of words rather than individual words. 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 hi' 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. Groups are the minimal meaningful units, not words. It is not appropriate to deal directly with words.

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. The L-Structure of a sentence shows all the potential word groups in the given sentence. It is very useful to identify these groups right in

the beginning since all subsequent steps in syntactic analysis can then view sentences as sequences of groups rather than as sequences of individual words. 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.

In UCSG we start by grouping words 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. A set of role assignments that simultaneously satisfies all the F 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. Let us now see how exactly linear, hierarchical and functional structure are determined taking examples from the Telugu language. In the process, we will get to see which aspects of knowledge are relevant and to what extent.

3 Linear Structure

In the linear structure analysis phase, we are concerned with the task of identifying word groups. Some examples of word groups in Telugu are

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 lexicon and carry out morphological analysis if required. As we keep looking up the words from left to right, we also build up the word groups. We take lexical ambiguities into consideration and obtain all possible word groups. We keep noting down useful grammatical features of the groups including surface case marking information which will be useful later for functional role assignment.

3.1 Lexicon and Morphology

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 its demands on the contents and structure of its lexicon. In UCSG the lexicon lists words and for each word gives information about grammatical categories and subcategorization, as also grammatical features 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/postpositions are marked for the functional roles they indicate and relative words and subordinating conjunctions are marked for the nature of the clauses they indicate. Content words may also show semantic features useful for functional role

assignment. Since we are only dealing with syntax, we do not need meanings, usages, etymology, pronunciation etc.

Whenever meaning is compositional and productive morphological rules exist, only the basic forms will 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. Morphology has several important roles to play. Morphology is used to recognize morphological variants of stored words. Morphological analysis also provides useful information for forming word groups. 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. Morphology has a much greater significance in relatively free word order languages than in positional languages. Telugu morphology is very rich and a single word may have hundreds and hundreds of morphological variants. See [6, 5] for more details on Telugu morphology.

3.2 Determining Linear Structure

The internal structure of word groups incorporates three aspects. Some items are optional. Some items can be repeatedly used. Also, the linear order of items is significant. Changing the word order renders the group ungrammatical or at least changes the meaning. This is true of both positional languages and relatively free word order languages.

In UCSG we identify word groups right in the beginning and treat these groups as atomic units throughout the rest of syntactic analysis. We do not intend to make detailed analysis of each of these groups. We have argued that it is not wise to attempt to capture hierarchical structure within word groups. Either there is no hierarchy of meaningful parts at all or there is a hierarchical structure but it is beyond syntax to capture that structure [2]. Further, 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 at all relevant for linear structure analysis. Our noun groups do not include relative clauses. In UCSG thus 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.

Linear structure implies constraints on linear order, repetition and optionality. No other types of constraints are relevant. Hence linear structure analysis can be carried out using finite state machines 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. In a single linear scan of the sentence we can obtain all the potential word groups. 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.

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. 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. Linear structure analysis requires only lexical and morphological level knowledge.

4 Hierarchical Structure

The H-Structure of a sentence depicts the various clauses, their nested relationships as well as the clause boundaries. Unlike linear structure, hierarchical structure 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. 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.

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, assignment of 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.

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. 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. In UCSG we show that it is possible to determine clause structure before and without applying the functional structure constraints.

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. 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. Some examples from Telugu are given below.

In general, every clause in every sentence would be well marked for one of its two boundaries by sentinels. There are parametric variations across languages as to which types of clause are marked for their start and end positions. In Telugu, clausal subjects, clausal objects, modifier clauses as well as relative clauses are all overtly marked by sentinels for their end positions. In general sentence boundaries also act as clause boundaries and are therefore similar to sentinels in their function.

If we ignore the participant 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 except the main 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. Verb group-sentinel sequences carry a lot of useful information about clause structure.

4.1 Determining Hierarchical Structure

To determine the hierarchical structure, we make use of the of the groups already obtained in the linear analysis phase. All the verb groups will have already been obtained in the linear analysis phase and so 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 mainly determined by the sequences of verb groups and sentinels. 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.

Sentinels 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. We will now see how exactly we can determine the local domains of clauses in Telugu sentences.

4.1.1 The Hierarchical Structure of Telugu Sentences

We make distinctions between three types of clauses in a sentence in Telugu. Let us name them f_clause, sub_clause and rel_clause. 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. Clausal subjects, clausal objects and the modifier clauses are all termed sub_clauses. Sub_clauses fill the participant roles. Relative clauses, named rel_clause, are modifiers of participants in a clause. This distinction between rel_clauses and sub_clauses is important. Sub_clauses end with the sentinel named sb. Rel_clauses end with the sentinel denoted rl. The f_clause is without any explicit sentinel. Thus every clause has one verb group, denoted vg, and every clause except the f_clause is marked by an ending sentinel, sb or rl.

With these things in mind we can now write down the H grammar rules for Telugu. A Telugu sentence s could be either the f_clause alone or f_clause with preceding sub_clauses. There can also be more than one sub_clause since sub_clauses include clausal 'subjects', clausal complements as well as modifier clauses. The f_clause itself would either be a vg alone or a vg with rel_clauses before the vg. A rel_clause has just a sentence s followed by the sentinel rl. A sub_clause is similar except that the sentinel would be sb. Hence the H-Structure rules for Telugu can be written as

Rule 1 s → (sub_clause)* f_clause

Rule 2 f_clause → (rel_clause)* vg

Rule 3 rel_clause → s rl

Rule 4 sub_clause → s sb

where the '*' indicates zero or more occurrences. 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 sequence. 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 the Telugu sentence

1)

baaMkunu doocaalanukonna doMgalu raatri aDavilooniki paaripooyi uMTaarani pooliisulu nammaaru

The input to the hierarchical structure analyzer is therefore the string 'vg rl vg sb vg' which corresponds to the sequence 'doocaalanuko ina uMTaaru ani nammaaru'. Observe that the 'sentence' we consider for hierarchical structure analysis - the verb group-sentinel sequence, is significantly smaller than the original sentence in size. The four rules above give us the following tree structure:

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

We will now specify how the clause boundaries can also be determined. In an ordered tree, the spans of a constituent are related to the spans of its child nodes. Note that our 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 terminal symbols 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 s spans right from the start of the sentence. Child nodes of a given node may span all the way from the start of the parent node or from the end position of a left sibling if any, whichever is later. Thus by propagating the left most limiting positions down the hierarchy, we can obtain the left ends of the grey areas. Thus determining the left end of the grey areas is also very simple. The result is shown below. Observe that for terminals the spans are exactly known and there are no grey areas. The i^{th} word in the sentence is taken to go from position i to position (i + 1). Also, where vg and sentinels are composed into a single orthographic word, we count the positions as if the groups were separate.

```
[(s,1,2,12),[(sub_clause,1,2,10),[(s,1,2,9),
  [(f_clause,1,2,9),[(rel_clause,1,2,4),[(s,1,2,3),
    [(f_clause,1,2,3),(vg,_,2,3)]],(rl,_,3,4)],(vg,_,7,9)]],
      (sb,_,9,10)],[(f_clause,10,11,12),(vg,_,11,12)]]
```

For convenience let us depict sub_clauses by curly brackets and rel_clauses by square brackets. Since there may be several clauses of a type in a given sentence, we attach numbers to brackets to help match the corresponding left and right brackets. 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

Note that the local domain of the sub_clause extends from the beginning of the sentence till the sb. The local domain of the rel_clause extends from the starting position in the sentence till the rl. Only the first word in the sentence is within the grey areas of these clauses. Thus to analyze the main clause we need to look at only the first word and the last two words in the sentence. We do not even have to consider the other words. In the same way, the functional structure analysis of the other two clauses are also localized. Localization of functional structure analysis makes UCSG very efficient.

The four rules given above form a complete set and are sufficient to deal with the hierarchical structure of clauses in Telugu. We thus have a very small and an extremely simple set of rules. 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. The main clause is always signalled by the last vg in a Telugu sentence. 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. Because of multiple possibilities for sub_clause attachment, we need to be able to produce all possible H-Structures. The recursive calls in rules 3 and 4 are sufficient to obtain all possible H-Structures. We only need these few rules and syntactic information about verb groups and sentinels to carry out hierarchical structure analysis. No other kind of knowledge is really essential.

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. CFGs are ideal only when both linear *and* hierarchical structure are significant and functional dependencies are not to 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.

Traditionally, the complexity of parsing is expressed taking the length of the sentence, n, as a basis. However complexity measures based on length of the input sentence tend to be misleading. Lengths of sentences in different languages vary significantly and systematically. Telugu sentences typically have much smaller number

of words than English sentences. Telugu is a morphologically rich language and whatever English accomplishes by grouping several words together, Telugu achieves the same 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.

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. Further, we have only a very small set of rules in UCSG. 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. The Paninian grammar does not address the issue of hierarchical structure of clauses at all. It simply analyzes the functional structure of the entire sentence in one go, losing the advantages of the localizing effect of clause structure analysis.

Further, we have shown [2] that both the depth of the H-Structure tree and the branching factor are bounded. Thus 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 be done, at least in principle, in linear time, that too linear in the number of verb groups and sentinels only.*

Hierarchical structure of clauses is the most crucial aspect of UCSG. Hierarchical structure grammar also happens to be the most universal aspect of grammar - the H 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.

5 Functional Structure

An F-Structure shows the functional roles played by all the participants in the predication. Both mandatory and optional roles are depicted. 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. 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.

5.1 Functional Roles

In UCSG, we have aimed at developing a small set of universal functional roles. Our case system is also influenced by considerations of computational viability and efficiency. In a computational system it should be possible to assign case roles automatically and without requiring human intuitions or judgements. The case role definitions may be semantic in nature but it must be possible to specify role assignments in almost purely syntactic terms. We have found that question-answer view helps a lot not only for designing a case system but also at all levels of theory and implementation in NLP [3]. See [4, 2] for more details. While questions and answers also do not always appear intuitive and natural, we believe that this view is generally much more intuitive in comparison with other approaches. Our set of cases is smaller and hence more general than many others proposed earlier.

We have eight roles named SUBJ, OBJ, OBJ2, SUBJ_QUAL, SPACE, TIME, ADVERBS and CAUSE_LINK. We do not make finer distinctions such as AGENT, CO-AGENT and EXPERIENCER; DONOR, RECIPIENT or BENEFICIARY. The SPACE role includes points in space, regions, origins, destinations, direction, etc. 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. We have not included an INSTRUMENT case role.

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, 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.

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.

Thus 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. In UCSG F-Structures are the final and complete structural descriptions. Hence we need only those pieces of knowledge that help us to obtain the appropriate F-Structures. 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.

5.3 Determining Functional Structure

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. 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. Formal techniques such as integer programming, Assignment Problem (as in Operations Research) and matching in bipartite graphs can be used [1].

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, ' 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 can 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. Global constraints if any can be checked at the end.

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 roles can be shared between two or more clauses or they may be completely missing. Relative clauses and indirect questions are examples of such syntactic constructions. To the extent that participants can be shared between clauses, functional structures of clauses are dependent on one another. However,

these interdependencies amongst clauses are limited. We notice that all the inter-clause dependencies are related to the hierarchical structure of clauses in a sentence. Each clause can be analyzed essentially in isolation if we work from the main clause down the clause hierarchy and information and expectations about missing and shared constituents is passed on as a parameter for the analysis of each clause.

We propose that the functional structure analysis start from the main clause. We start our search by attempting to assign the mandatory and optional roles to those constituents which lie within the specified local domain of the main clause. 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. The sentinels themselves provide useful clues about which role is missing or shared. We pass down information about, 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 somethings 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. To analyze a clause, we need to consider only those groups falling within the local domain of a given clause and only those roles as required or permitted by the subcategorization frame of the verb group in that clause. This localization makes the functional structure analysis very efficient. We have seen that linear structure analysis is linear in time complexity and complexity of hierarchical structure analysis is cubic in the number of verb groups and sentinels even without any assumptions about the finiteness of H-Structures. Thus UCSG is more efficient on the whole compared to other available grammar formalisms.

6 The UCSG Telugu Parser

The parser has been developed in Arity Prolog version 6.1 to run on personal computers running the DOS operating system. The parser opens up windows, issues title and start up messages and reads in the required data files. The H rules, written in the CFG format, are read in from a data file and the rules are analyzed to obtain useful information. 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. For unknown words, the sys-

tem can obtain the required information from the user. 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.

Screen dumps depicting the parser output for a sample Telugu sentence are included at the end. 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 parser displays functional structure clause by clause and shows the functional roles in fixed locations on the screen for visual impact and more effective communication. 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. The GIST script processor card is used for viewing in Telugu script.

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. Verb-less sentences are parsed by hypothesizing a default verb of the 'BE' type. In sentences with missing SUBJ, the morphological inflections on the vg provide required information for determining the implied SUBJ. The current version does not handle sentences with coordinate conjunctions or comparatives. Efforts are on to enhance the coverage and robustness. The parser is currently being applied to research in spelling and grammatical error detection and correction, metaphor interpretation and machine translation from English to Telugu.

7 Conclusions

In the past, proof or belief that grammars of a particular type are inadequate to deal with the syntax of human languages was simply taken to imply that more powerful grammars must be employed. In UCSG we instead ask which aspects of syntax require which type of grammar. By dividing the problem of syntactic analysis accordingly, we have made the grammar simpler and easier to write and the parsing process very efficient. An appropriate way of dividing into modules has also helped us to see the underlying similarities and differences between different classes of human languages. Grammars written in the UCSG formalism tend to be more language invariant.

The division of syntactic analysis into L, H and F modules also helps us to clearly

understand the relative significance of different kinds of knowledge for syntactic analysis. Parsing in UCSG depends mostly on lexical, morphological and syntactic knowledge of the language. We do not use knowledge of the discourse context. We use a limited amount of semantic knowledge encoded in the form of semantic features associated with lexical items. We do not use any knowledge about the real world directly. The three modules of UCSG make use of the available knowledge selectively. For example, while the functional structure analyzer benefits from the knowledge of the actual words used as sentinels, the hierarchical structure analyzer only makes distinction between sentinels marking sub_clauses and sentinels marking rel_clauses. While the linear structure analyzer and the hierarchical structure analyzer find it sufficient to note that a word is a verb, the functional structure finds it useful to know the subcategorization details. The success of UCSG lies in applying only those pieces of knowledge which are really essential at each step. Contrary to what many others seem to suggest, UCSG shows that we can parse sentences in natural language without calling for large or unbounded amounts of semantic, pragmatic and general world knowledge.

References

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

[2] K Narayana Murthy. *Universal Clause Structure Grammar*. PhD thesis, Department of Computer and Information Sciences, University of Hyderabad, 1995.

[3] 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.

[4] 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.

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

[6] 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.

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