AUGMENTED TRANSITION NETWORK GRAMMARS FOR NATURAL LANGUAGE ANALYSIS

W.A. Woods

Rapporteur: Dr. J.M. Rushby

Abstract

The use of augmented transition network grammars for the analysis of natural language sentences is described. Structure-building actions associated with the arcs of the grammar network allow for the re-ordering, restructuring and copying of constituents necessary to produce deep-structuring representations of the type normally obtained from a transformational analysis, and conditions on the arcs allow for a powerful selectivity which can rule out meaningless analyses and take advantage of semantic information to guide the parsing. The advantages of this model for natural language analysis are discussed in detail and illustrated by examples. An implementation of an experimental parsing system for transition network grammars is briefly described.

1. Preface

Late in the fall of 1968, in order to provide mechanical input for a semantic interpreter, I began constructing a parsing program based on the notion of a recursive transition network grammar, a model very much like a finite-state transition graph except for the presence of non-terminal as well as terminal symbols and labels on the arcs. A non-terminal label causes a recursive application of the transition network to recognize a construction of the type indicated by the label before the transition so labelled is permitted. This model, which is weakly equivalent to a non-deterministic pushdown store automaton, occurred to me as a natural representation of the type of grammar that one would get if one carried the use of the Kleene * operator and bracketed alternatives in the right-hand sides of context-free grammar rules (a notation used by many linguists) to its logical conclusion by permitting arbitrary regular expressions as the right-hand sides of rules. One could then merge all of the rules with a given non-terminal symbol as their left-hand side and could represent this rule either by its regular expression or alternatively by an equivalent finite state transition graph (over the total vocabulary of terminal and non-terminal symbols). It is this latter form of representation which I have called a recursive transition network. In the course of this implementation, I learned that a similar approach to natural language analysis had been used by Thorne, Bratley and Dewar (1968) and by Bobrow and Fraser (1969). My approach is in effect a generalization and formalization of these earlier parsers and provides a number of additional capabilities.

In addition to many advantages for efficient context-free recognition and improved strong generative power, the transition network model also provides a convenient means for incorporating syntactic and semantic conditions for guiding the parsing and for performing transformations and relocations of constituents. This is done by associating arbitrary conditions and structure building actions with the arcs of the network. This augmented network is a kind of "transducer", whose effects are to make changes in the contents of a set of registers associated with the network and whose transitions can be conditional on the contents of those registers. Registers can be used to hold pieces of syntactic structure whose position and function in the syntactic structure being built might not yet have been determined.

Experience with the parsing system has shown it to be an extremely powerful system -- capable of performing the equivalent of transformational analysis in little more time than that customarily required for context-free analysis alone. In addition, the system is convenient for the designer of the grammar and facilitates experiments with various types of structural representations and various parsing strategies.

1.1 Motivation

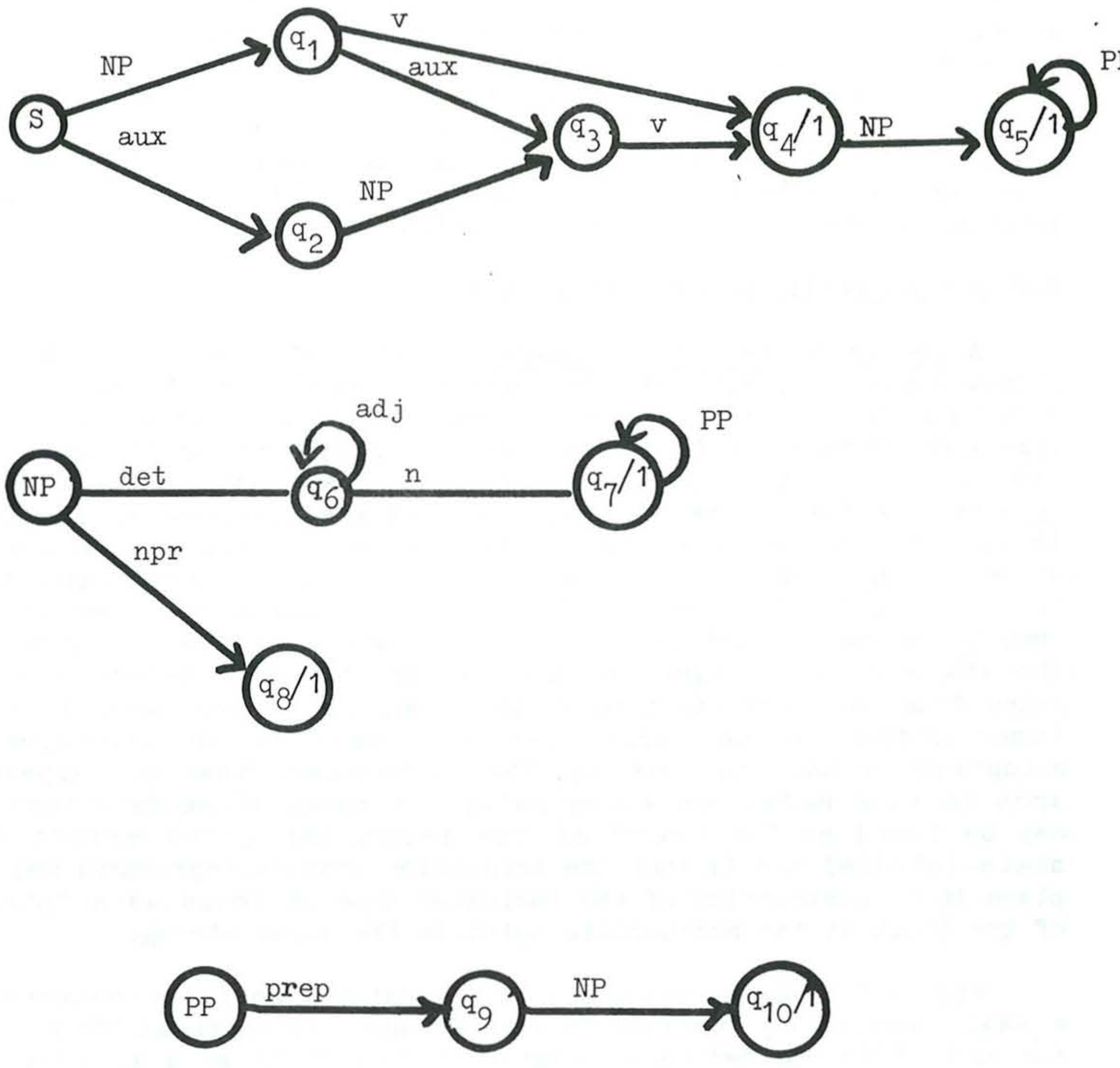
One of the early models for natural language grammars was the finite-state transitiongraph corresponding to a finite-state machine that accepted (or generated) the sentences of a language. In this model, the grammar is represented by a network of nodes and directed arcs connecting them. The nodes correspond to states in a finite-state machine, and the arcs represent transitions from state to state. Each arc is labelled with a symbol whose input can cause a transition from the state at the tail of the arc to the state at its head. This model has the attractive feature that the sequences of words which make up a sentence can be read off directly by following the paths through the grammar from the initial state to some final state. Unfortunately, the model is grossly inadequate for the representation of such grammars because of its failure to capture many of the regularities of such grammars. The most notable of these is the pushdown mechanism that permits one to suspend the processing of a constituent at a given level while using the same mechanism to process an embedded constituent.

Suppose, however, that one added the mechanism of recursion directly to the transition graph model by fiat. That is, suppose that one took a collection of transition graphs each with a name, and permitted not only terminal symbols to be labels on the arcs but also non-terminal symbols naming constructions which must be present in order for the transition to be followed. The determination of whether such a construction was in fact present in a sentence would be done by a "subroutine call" to another transition graph (or the same one). The resulting model of grammar, which we will call a recursive transition network, is equivalent in generative power to that of a context-free grammar or pushdown store automaton, but as we will show allows for greater efficiency of expression, more efficient parsing algorithms, and natural extension by "augmentation" to more powerful models which allow various degrees of context dependence and more flexible structure-building during parsing. We will argue in fact that an "augmented" recursive transition network is capable of performing the equivalent of transformational recognition without the necessity of a separate inverse transformational component, and that this parsing can be done in an amount of time which is comparable to that or ordinary context-free recognition.

1.2 Recursive transition networks

A recursive transition network is a directed graph with labelled states and arcs, a distinguished state called the start state, and a distinguished set of states called final states. It looks essentially like a non-deterministic finite state transition diagram except that the labels on the arcs may be state names as well as terminal symbols. The interpretation of an arc with a state name as its label is that the state at the end of the arc will be saved on a pushdown store and that control will jump (without advancing the input tape) to the state that is the arc label. When a final state is encountered then the pushdown store may be "popped" by transferring control to the state which is named on the top of the stack and removing that entry from the stack. An attempt to pop an empty stack when the last input character has just been processed is the criterion for acceptance of an input string. The state names that can appear on arcs in this model are essentially the names of constructions that may be found as "phrases" of the input tape. The effect of a state-labelled arc is that the transition that it represents may take place if a construction of the indicated type is found as a "phrase" of the input at the appropriate point in the input string.

Figure 1 gives an example of a recursive transition network for a small subset of English. It accepts such sentences as "John washed the car", "Did the red barn collapse?", etc. It is easy to visualize the range of acceptable sentences from inspection of the transition network. To recognize the sentence, "Did the red barn collapse", the network is started in state S. The first transition is the aux transition to state q_2 permitted by the auxiliary "did". From state q_2 we see that we can get to state q_3 if the next "thing" in the input string is an NP. To ascertain if this is the case, we call the state NP. From state NP we can follow the arc labelled det to state q_6 because of the determiner "the". From here, the adjective "red" causes a loop which returns to state q_6, and the subsequent noun "barn" causes a transition to state q_7. Since state q_7 is a final state, it is possible to "pop up" from the NP computation and continue the computation of the top level S, beginning in state q_3 which is at the end of the NP arc. From q_3 the verb "collapse" permits a transition to the state q_4, and since this state is final and "collapse" is the last word in the string, the string is accepted as a sentence.



S is the start state

q_4, q_5, q_7, q_8 and q_{10} are the final states

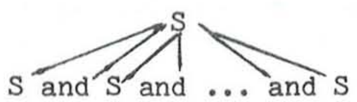
Figure 1 : A sample transition network

The fact that the recursive transition network is equivalent to a pushdown store automaton is not difficult to establish. Every recursive transition network is essentially a pushdown store automaton whose stack vocabulary is a subset of its state set. The converse fact that every pushdown store automaton has an equivalent transition net could be established directly, but can be more simply established by noting that every pushdown store automaton has an equivalent context-free grammar which has an equivalent recursive transition net as we will show.

1.3 Augmented transition networks

It is well known (cf. Chomsky 1964) that the strict context-free grammar model is not an adequate mechanism for characterizing the subtleties of natural languages. Many of the conditions which must be satisfied by well-formed English sentences require some degree of agreement between different parts of the sentence which may or may not be adjacent (indeed which may be separated by a theoretically unbounded number of intervening words). Context-sensitive grammars could take care of the weak generation of many of these constructions, but only at the cost of losing the linguistic significance of the "phrase structure" assigned by the grammar (cf. Postal 1964). Moreover, the unaided context-free grammar model is unable to show the systematic relationship that exists between a declarative sentence and its corresponding question form, between an active sentence and its passive, etc. Chomsky's theory of transformational grammar (Chomsky 1965), with its distinction between the surface structure of a sentence and its deep structure, answers these objections but falls victim of inadequacies of its own (cf. Schwarcz 1967, or McCawley 1968). In this section we will describe a model of grammar based on the notion of a recursive transition network which is capable of performing the equivalent of transformational recognition without the need for a separate transformational component, and which meets many of the objections that have been raised against the traditional model of transformational grammar.

The basic recursive transition network model as we have described it is weakly equivalent to the context-free grammar model and differs in strong equivalence only in its ability to characterize unbounded branching, as in structures of the form :



The major features which a transformational grammar adds to those of the context-free grammar are the abilities to move fragments of the sentence structure around (so that their positions in the deep structures are different from those in the surface structure), to copy and delete fragments of sentence structure, and to make its actions on constituents generally dependent on the contexts in which those constituents occur. We can add equivalent facilities to the transition network model by adding to each arc of the transition network an arbitrary condition which must be satisfied in order for the arc to be followed, and a set of structure building actions to be executed if the arc is followed. We call this version of the model an <u>augmented</u> transition network.

The augmented transition network builds up a partial structural description of the sentence as it proceeds from state to state through the network. The pieces of this partial description are held in <u>registers</u> which can contain any rooted tree or list of rooted trees, and which are automatically pushed down when a recursive application of the transition network is called for, and restored when the lower level (recursive) computation is completed. The structure-building actions on the arcs specify changes in the contents of these registers in terms of their previous contents, the contents of other registers, the current input symbol, and/or the results of lower level computations. In addition to holding pieces of substructure that will eventually be incorporated into a larger structure, the registers may also be used to hold flags or other indicators to be interrogated by conditions on the arcs.

Each final state of the augmented network has associated with it one or more conditions which must be satisfied in order for that state to cause a "pop" - i.e. to return from a lower level computation to the next higher one, or to complete the analysis when the end of the string is encountered. Paired with each of these conditions is a function which computes the value to be returned by the computation. A distinguished register denoted by *, which contains the current input word when a word is being scanned, is set to the result of the lower level computation when the network returns to the arc which called for the recursive computation.

1.3.1 Representation of augmented newtorks

To make the discussion of augmented transition networks more concrete, we give in figure 2 a specification of a language in which an augmented transition network can be represented. The specification is given in the form of an extended context-free grammar in which a vertical bar separates alternative ways of forming a construction and the Kleene star operator (*) is used as a superscript to indicate arbitrarily repeatable constituents. The non-terminal symbols of the grammar consist of English descriptions enclosed in angle brackets, and all other symbols except the vertical bar and the superscript * are terminal symbols (including the parentheses, which indicate list structure). The * which occurs as an alternative right-hand side for thhee rule for the construction form , however, is a terminal symbol and is not to be confused with the superscript *'s which indicate repeatable constituents. The first line of the figure says that a transition network is represented by a left parenthesis, followed by an arc set, followed by any number of arc sets (zero or more), followed by a right parenthesis. An arc set in turn consists of a left parenthesis, followed by a state name, followed by any number of arcs, followed by a right parenthesis, and an arc can be any one of the four forms indicated in the third rule of the grammar. The remaining rules are interpreted in a similar fashion.

```
<transition network> → (<arc set> <arc set>*)

<arc set> →  (<state> <arc>*)

<arc>  →  (CAT <category name> <test> <action>* <term act>)|

          (PUSH <state> <test> <action>* <term act>)|

          (TST <arbitrary label> <test> <action>* <term act>)|

          (POP <form> <test>)

<action> → (SETR <register> <form>)|

           (SENDR <register> <form>)|

           (LIFTR <register> <form>)

<term act> → (TO <state>)|

             (JUMP <state>)

<form> → (GETR <register>)|

          *    |

          (GETF <feature>)|

          (BUILDQ <fragment> <register>*)|

          (LIST <FORM>*)|

          (APPEND <form> <form>)|

          (QUOTE <arbitrary structure>)
```

Figure 2 : Specification of a language
for representing augmented transition networks

execute the actions: (SETR AUX *), which puts the current word "does" into a register named AUX, (SETR TYPE (QUOTE Q)), which puts symbol "Q" into a register named TYPE, and (TO Q2), which causes the network to enter state Q2 scanning the next word of the sentence "John".

2. State Q2 has only one arc leaving it, which is a push to state NP/. The push will be successful and will return a representation of the structure of the noun phrase which will then become the value of the special register *. We will assume that the representation returned is the expression "(NP John)". Now, having recognized a construction of type NP, we proceed to perform the actions on the arc. The action (SETR SUBJ *) causes the value "(NP John)" to be placed in the register SUBJ, and the action (TO Q3) causes us to enter the state Q3 scanning the next word "like". The register contents at this point are:

TYPE	:	Q
AUX	:	does
SUBJ	:	(NP John).

3. From state Q3, the verb "like" allows a transition to state Q4, setting the contents of a register V to the value "like" in the process, and the input pointer is advanced to scan the word "Mary".

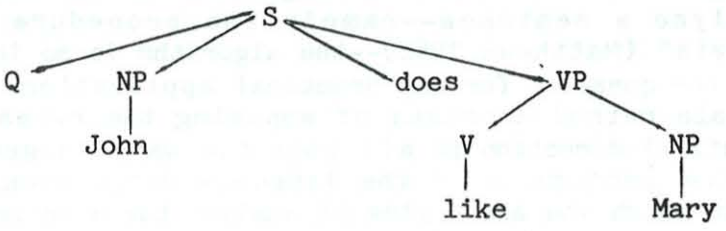
4. Q4, being a final state could choose to "POP", indicating that the string that has been processed so far is a complete sentence (according to the grammar of figure 1); however, since this is not the end of the sentence, this alternative is not succcessful. However, the state also has an arc which pushes down to state NP/, and this alternative will succeed, returning the value "(NP Mary)". The action (SETR VP (BUILDQ (VP (V +) *) V)) will now take the structure fragment "(VP (V +) *)" and substitute the current value of * for the occurrence of a * in the fragment and replace the occurrence of + with the contents of the indicated register V. The resulting structure, "(VP (V like) (NP Mary))" will be placed in the register VP, and the action (TO Q5) causes a transition to state Q5 scanning beyond the end of the input string. The register contents at this point are:

TYPE	:	Q
AUX	:	does
SUBJ	:	(NP John)
V	:	like
VP	:	(VP (V like) (NP Mary))

5. We are now scanning the end of the sentence, and since Q5 is a final state (i.e., it has a "POP" arc), and the condition T is satisfied, the sentence is accepted. The form "BUILDQ (S + + + +) TYPE SUBJ AUX VP)" specifies the value to be returned as the analysis of the sentence. The value is obtained by substituting the contents of the registers TYPE, SUBJ, AUX, and VP for the successive instances of the symbol "+" in the fragment "(S + + + +)" to give the final sentence analysis:

   (S Q (NP John) does (VP (V like) (NP Mary)))

   which represents the parse tree:



In ordinary context-free recognition, the structural descriptions of sentences are more or less direct representations of the flow of control of the parser as it analyzes the sentence. The structural descriptions assigned by the structure building rules of an augmented transition network, as we can see from the example, are comparatively independent of the flow of control of the algorithm.

This is not to say that they are not determined by the flow of control of the parser, for this they surely are; rather we mean to point out that they are not isomorphic to the flow of control as in the usual context-free recognition algorithms. It is possible for a constituent that is found in the course of analysis to appear in the final structural description several times or not at all, and its location may be entirely different from that in which it was found in the surface structure. In addition, the structural description assigned to a constituent at one point during the analysis may be changed or transformed before that structure is incorporated into the final structural description of the sentence as a whole. These facilities plus the ability to test arbitrary conditions allow the equivalent of a transformational deep structure to be constructed while the parser is performing transitions that are isomorphic to the surface structure of a sentence.

1.4 Transformational recognition

The usual model of transformational grammar is a generative model consisting of a context-free (base) grammar and a set of transitional rules which map syntax trees into new (derived) syntax trees. The generation of a sentence with such a grammar consists of first constructing a deep structure using the base component grammar and then transforming this deep structure into a surface structure by successive applications of transformations. The terminal nodes (or leaves) of the surface structure tree give the final form of the sentence. This model of transformational grammar is totally oriented toward the generation of sentences rather than their analysis, and although there is clearly an algorithm for the use of such a grammar to analyze a sentence--namely the procedure of "analysis by synthesis" (Matthews 1962)--the algorithm is so inefficient as to be out of the question for any practical application. (The analysis by synthesis method consists of applying the rules in the "forward" (generative) direction in all possible ways to generate all of the possible sentences of the language while looking to see if the sentence which you are trying to analyze turns up in the list.)

Two attempts to formulate more practical algorithms for transitional recognition (Petrick 1965, and MITRE 1964) resulted in algorithms which were still too time-consuming to be practical for the analysis of more than a few test sentences with small sample grammars. Both of these algorithms attempt to analyze sentences by applying the transformations in reverse, a procedure which is far less straightforward than it sounds. The difficulty with simply performing the transformations in reverse is twofold. First, the transformations operate on tree structures and produce tree structures as their values. In the forward direction, they begin with the deep structure tree and end with the surface structure tree. In order to reverse this process, one needs first to obtain a surface structure tree for the input sentence. However, there is no component in the transformational model which characterizes the possible surface structures (their only characterization is implicit in the changes which can be made in the deep structures by means of the transformations). Both the MITRE and the Petrick analysis procedures solve this problem by constructing an "augmented grammar" which consists of the rules of the original base component grammar plus additional rules which characterize the structures which can be added by transformations. In the MITRE procedure this "surface grammar" is constructed by hand and no formal procedure is available for constructing it from the original transformational grammar. In the Petrick procedure, there is a formal procedure for obtaining an augmented grammar but it will not necessarily terminate unless the length of the possible input sentences is first circumscribed (which unfortunately reduces the class of sentences that can be accepted to a finite set--theoretically analyzable by table lookup).

In the MITRE procedure, the augmented grammar is used to assign a complete "tentative" surface structure which is then subjected to inverse transformations. In the Petrick procedure, inverse transformations are applied to partially built up surface structures and the processes of applying transformations and building structure are interwoven. In both systems, the inverse transformations may or may not produce a legitimate deep structure. If they do, then the sentence is accepted, but if they do not, then the tentative surface structure was spurious and is rejected. There is no way to construct a context free surface grammar which will assign all and only legitimate surface structures. One must settle for one which will assign all legitimate surface structures plus additional spurious ones. Moreover, the only way to tell the two apart is to perform the inverse transformations and check the resulting "tentative" deep structures.

The second difficulty in this method of analysis is the combinatorial explosion of the number of possible inverse transformation sequences that can be applied to a given surface structure tree. Although many of the transformations when applied in the forward direction are obligatory, so that only one possible action can be taken, almost all of the inverse transformations are optional. The reason for this is that even though a given structure looks like it could have been produced by a given forward transformation so that the inverse transformation can be performed, there is no guarantee that the same structure could not have arisen in a transformational derivation in some other way. Therefore both the alternative of applying the inverse transformation and that of not applying it must both be tried whenever an inverse transformation can apply. The number of active paths can grow exponentially with the number of transformations applied. Moreover, the forward transformations usually don't specify much information about the structure which results from applying the transformation (even though the linguist may know a good deal about what the resulting structure must be like). For this reason the inverse transformations are not as selective as their forward counterparts and many more spurious applications of transformations are allowed. That is, whereas most forward sequences of transformations will lead to successful surface structures, most inverse sequences will not lead to legitimate deep structures, and a large amount of unnecessary wasted effort is therefore expended on dead end paths. To make matters worse, it is not always clear what the stopping conditions on the inverse transformational process should be. Some inverse transformational sequences could go on forever and it is not clear what set of conditions is sufficient to guarantee that a given sequence will not eventually lead to a legitimate deep structure. In short, the inverse transformational process is an extremely complicated one and is impractically inefficient to implement.

1.5 <u>Augmented transition networks for transformational recognition</u>

Kuno (1965) suggested that it should be possible to augment the surface structure grammar of a transformational grammar in such a way that it "remembered" the equivalent deep structure constructions and could build the deep structure of the sentence while doing the surface structure parsing, without the necessity of a separate inverse transformational component. The model which he proposed at that time, however, was not adequate to deal with some of the more powerful transformational mechanisms such as the extraposition of a constituent fom an arbitrarily deep embedding. The augmented transition network, on the other hand, provides a model which is capable of doing everything that a transformational grammar can do and is therefore a realization of part of the Kuno prediction. It remains to be seen whether a completely mechanical procedure can be developed to take a transformational grammar in the usual formalism and translate it into an equivalent augmented transition network, but even if such a procedure is available, it may still be more approppriate to use the transition network model directly for the original linguistic research and grammar development. The reasons for this are several: first, the transition network that could be developed by a mechanical procedure from a traditional transformational grammar could not be expected to be as efficient as that which could be designed by hand. Moreover, the transition network model provides a mechanism which satisfies many of the objections which have been raised by linguists against the transformational grammar as a linguistic model (such as its incompatibility with many psycholinguistic facts which we know to characterize human language performance).

A third reason for preferring the transition network model to the usual formulation of transformational grammar is the power which it contains in its arbitrary conditions and its structure building actions. The model is equivalent to a Turing machine in power and yet the actions which it performs are "natural" ones for the analysis of language. Most linguistic research in the structure of language and mechanism of grammar has attempted deliberately to build models which do not have the power of a Turing machine but which make the strongest possible hypotheses about language mechanisms by proposing the least powerful mechanism that can do the job. As a result of this approach many variations of the transformational grammar model have been proposed with different basic repertories of transformational mechanisms. Some have cyclic transformation rules, others do not; some have a distinct "post cycle" that operates in a different mode after all of the cyclic rules have been applied. There are various types of conditions that may be asked, some have ordered rules, some have obligatory rules, some have blocking rules, etc. In short there is not a single transformational grammar model, there are myriads. Moreover these models are more or less incomparable. They do not fall within a single general framework so that their relative merits can be compared. If one such model can handle some features of language and another can handle different features, there is no systematic procedure for incorporating them both into a single model. In the augmented transition network model, the possibility exists to add to the model whatever facility is needed and seems natural to do the job. One can add a new mechanism by simply inventing a new basic predicate to use in conditions or a new function to use in the structure building rules. It is still possible to make strong hypotheses about the types of conditions and actions that are required, but when one finds that he needs to accomplish a given task for which his basic model has no "natural" mechanism, there is no problem in extending the augmented transition network model to include it. This requires only the relaxation of the restrictions on the types of conditions and actions, and no reformulation of the basic model.

References

Bates, M. (1978). "The Theory and Practice of Augmented Transition Networks", in L. Bolc (Ed.) Natural Language Communication with Computers, Springer, Berlin.

Bobrow, D.G. and Fraser, J.B. (1969). "An Augmented State Transition Network Analysis Procedure", Proc. Internat. Joint Conf. on Artificial Intelligence, Washington D.C., pp. 557-567.

Burton, R. and Woods, W.A. (1976). "A Compiling System for Augmented Transition Networks", preprints of 6th International Conference on Computational Linguistics (COLING 76), June, Ottawa, Canada.

Chomsky, N. (1964). "A Transformational Approach to Syntax", in The Structure of Language, J.A. Fodor, J.J. Katz (Eds.), Prentice-Hall, N.J.

Chomsky, N. (1965). "Aspects of the Theory of Syntax", MIT Press, Cambridge, Mass.

Kuno, S. (1965). "A System for Transformational Analysis", Report NSF-15, Harvard Computing Lab., Cambridge, Mass.

McCarthy, J. et al. (1962). "LISP 1.5 Programmer's Manual", MIT Computing Centre, Cambridge, Mass.

McCawley, J.D. (1968). "Meaning and the Description of Languages", in Kotoba No Ucho, TEC Co. Ltd., Tokyo.

Matthews, G.H. (1962). "Analysis by Synthesis of Natural Languages", Proc. 1961 Internat. Conf. on Machine Translation and Applied Language Analysis, HMSO, London.

MITRE (1964). "English Pre-processor Manual", Report SR-132, The Mitre Corp., Bedford, Mass.

Petrick, S.R. (1965). "A Recognition Procedure for Transformational Grammars", Ph.D. Thesis, Dept. Modern Languages, MIT, Cambridge, Mass.

Postal, P.M. (1964). "Limitations of Phrase Structure Grammars", in The Structure of Language, J.A. Fodor, J.J. Katz (Eds.), Prentice-Hall, N.J.

Schwarcz, R.M. (1967). "Steps toward a Model of Linguistic Performance: A Preliminary Sketch", Mechanial Translation, Vol. 10, pp. 39-52.

Thorn, J., Bratley, P. and Dewar, H. (1968). "The Syntactic Analysis of English by Machine", in Machine Intelligence 3, D. Michie (Ed.), American Elsevier, N.Y.

Woods, W.A. (1969). "Augmented Transition Networks for Natural Language Analysis", Report No. CS-1, Aiken Computation Laboratory, Harvard University, December. (Available from ERIC as ED-037-733; also from NTIS as Microfiche PB-203-527).

Woods, W.A. (1970). "Transition Network Grammars for Natural Language Analysis", CACM, Vol. 13, No. 10, October.

Woods, W.A. (1973). "An Experimental Parsing System for Transition Network Grammars", presented at the Symposium in Natural Language Processing, Courant Institute of Mathematical Sciences, New York University, December. In Natural Language Processing, R. Rustin (Ed.), Algorithmics Press. (Also BBN Report No. 2362, May 1972).

Woods, W.A. (1975). "Syntax, Semantics and Speech", in D.R. Reddy (Ed.), Speech Recognition : Invited Papers at the IEEE Symposium, New York: Academic Press. (Also as BBN Report No. 3067).

Woods, W.A. (1977). "A Personal View of Natural Language Understanding", SIGART, Special Issue, February.

Woods, W.A. (1978). "Semantics and Quantification in Natural Language Question Answering", in Advances in Computers, Vol. 17. Academic Press, New York. (Also Report No. 3687, Bolt Beranek and Newman Inc., 1977).

Woods, W.A. (1979). "Semantics for a Question Answering System", New York: Garland Publishing Inc.

Woods, W.A. (1980). "Cascaded ATN Grammars", in American Journal of Computational Linguistics, Vol. No. 1, January-March 1980.

Woods, W.A., Kaplan, R.M. and Nash-Webber, B.L. (1972). "The Lunar Sciences Natural Language Information System: Final Report", BBN Report No. 2378, Bolt Beranek and Newman Inc., Cambridge, Ma., June. Available from NTIS as N72-28984).